خطوة بخطوة مع

Microsoft

Visual Studio 2008

فيجوال ستوديو 2008

C# And VB.net

اعداد

أحمد جمال خليفة

Designed by
Norhan

خطوة بخطوة مع

فيجوال ستوديو 2008

Microsoft

Visual Studio 2008

C# And VB.net

أحمد جمال خليفة

تقديم الكتاب

بقلم الأستاذ تركي العسيري

استطيع أن أقول انك شخص محظوظ جدا وحماتك فعلا تحبك، حيث انك ستتعلم برمجة NET Framework. من مبرمج (حقيقي) وليس مبرمج أكاديمي نطاق انجازاته لا تتعدى كتابة برامج ببضعة سطور لحل أسئلة اختبار أو واجب عملي، فانا أتحدث عن أحمد جمال وهو من كبار المحترفين العرب والذي له بصماته في مشاريع هي تتحدث عنه، ويكفينا فخرا بأنه المسئول الأول لموقع vb4arab والذي يمثل اكبر مجتمع للمبرمجين العرب حول العالم.

ما يعجبني كثيرا في أسلوب احمد جمال دقته في اختيار الكم المعرفي للتحدث عنه، خاصة وسط هذا الكم العددي الهائل من الانفجار المعلوماتي وكثرة المستندات، فهو يعرف تماما ما يحتاجه المبرمج وما لا يحتاجه، ويعلم جيدا من أين تؤكل الكتف وكيف ترمى العظام، إما عن جواهره (اقصد شفراته المصدرية Source Codes) فهو يتفنن ويبدع في اختيار الأمثلة التي يعرضها بحيث تغطي الكثير من الحالات الحقيقية Real World Cases والتي ستظهر للمبرمج عاجلا أم أجلا، ولا يكتفي بذكر أمثلة تشرح الفكرة فقط.

بخصوص الكتاب، فتقول إحدى الحكم الطريفة: "من الصعب أن تجد قطة سوداء.. في قاعة صماء .. وخصوصا إن لم يكن هنالك قطة خير شرا!" وكمحاولة لربطها بموضوعنا يمكن أن نقول انه "من الصعب أن تجد أفضل لغة برمجة (موجهة لـ NET.) في ظل هذا الزخم من اللغات .. وخصوصا إن لم يكون هنالك أي لغة برمجة خير شرا!" فمن بعد ما كانت لغات البرمجة لغات حقيقية مستقلة لها سماتها وشمائلها، أصبح الحديث في هذه الأيام عن تقنيات (Java، NET.، COM، DirectX، WPF، LINQ...)، وتكاد أن تكون لغات البرمجة شبيهة فيما بينها إلى حد كبير، وهذا الكتاب خير دليل على صحة ما رميت به، فكاتب هذا الكتاب –بناء على مقدمته- اعتمد على مجموعة كتب تتحدث بلغات مختلفة (VB،C#، ...الخ) ليتمكن من كتابة كتاب حول تقنية NET.، ليس هذا فقط بل يقدم شرح الشفرات المصدرية بأكثر من لغة!

ومن هذا المنطلق كان الاختيار (الذكي جدا) من الكاتب لعنوان الكتاب Visual Studio 2008 دون تحديد اللغة، فلا يوجد بعد اليوم مصطلح اسمه مبرمج Basic أو مبرمج C#، فقد توحدنا (نحن معشر المبرمجين) وأصبحنا جميعا ننتمي إلى عائلة " آل مبرمجي.NET".

يغطي هذا الكتاب ابرز المواضيع الأساسية لتنقية.NET، ومن ثم يأخذ بك جولة إلى مواضيع متفرقة مثل برمجة قواعد البيانات Database Programming، نماذج Windows Forms، وبرمجة ويب. كما سيسيل الكاتب لعابك أكثر عندما تعلم انه لم يتجاهل التقنيات الجديدة (والتي ظهرت مع الإصدار الثالث من إطار العمل NET Framework) مثل تقنية LINQ الموجه للتعامل المتقدم مع البيانات Data Entities، وتقنية WPF والتي تمثل

الجيل القادم لبرمجة واجهات الاستخدام قابلة للنقل Portable User Interfaces، وتقنية WCF والتي توفر حلولا أكثر إنتاجية للتطبيقات الموزعة Distributed Applications. وهذه مواضيع من النادر جدا أن يستطيع شخص دمجها في كتاب واحد.

أخيرا، قد تكون ارتكبت خطأ في قراءة هذا التقديم (الذي لا يسمن ولا يغني من جوع ولن يوفي حق الكتاب ومؤلفه)، لذلك يكفي ما أضعته من وقتك الثمين معي واستعن بالله ثم ابدأ بقلب الصفحة لتدخل عالم برمجة.NET من أوسع أبوابها.

-- تركي العسيري

al-asiri.COM

الفهرس

المحتويات

الباب 07 مكونات اللغة الرئيسية في net. – الجزء الثاني

الباب 08 مقدمة إلى البرمجة كائنية التوجه OOP

الباب 09 Interfaces – الواجهات

الباب 10 الأخطاء واقتناصها

الباب 11 التجميعات Collections

الباب 12 البرمجة المتقدمة في 2008 net.

الباب 14 برمجة النوافذ في ال net. Windows Forms

الباب 15 GDI+

الباب 16 Windows Presentation Foundation

الباب 18 LINQ

الباب 19 WWF

الباب 20 WCF

مقدمة الكتاب

بسم الله الرحمن الرحيم، والصلاة والسلام على أشرف الأنبياء والمرسلين، نبينا محمد وعلى آله وصحبه أجمعين.

اللهم لا سهل إلا ما جعلته سهلاً وأنت تجعل الحزن سهلاً .

أما بعد ...

فهذا الكتاب هو عبارة عن تجميع لسلسلة من الدروس استمرت على مدار ستة أشهر على منتدى فيجوال بيسك للعرب حول تقنيات .net 2008 الجديدة ، تم الانتهاء منها بحمد الله في أول سبتمبر 2008 ، ومن ثم تم تجميعها مع وضع إضافات جديدة عليها ضمن كتاب منسق هو الكتاب الذي تجده بين يديك اليوم .

فكرة الكتاب جاءت بعد قراءة كتاب Pro C# 2008 and the .net 3.5 Platform للمؤلف Andrew Troelsen، حيث وجدت مرجعاً كاملاً باللغة الإنجليزية لكل ما يختص بنسخة C# الجديدة من مايكروسوفت ، ومع أني لا أدعي أني قمت بعمل مرجع باللغة العربية إلا أنني أستطيع أن أزعم أني غطيت بصورة موجزة أغلب النقاط الرئيسية في مجال ال net. بلغتيه C# و VB.net ، وللمزيد أحلتك إلى روابط من MSDN لتستطيع التعمق في كل مجال على حدة .

في هذا الكتاب ربما لن تجد تفصيلاً شاملاً لنقطة ما ، فلا تنتظر مني مثلاً أن تعرف بعد نهاية فصل ما كل ما يتعلق بأحد الأدوات، ولكنك تستطيع التعرف على أكبر قدر من المعلومات حول بنية هذه الأداة وطريقة تعاملها وسلوكها داخل برنامجك ، فهذا الكتاب يهدف لأن تستطيع من خلاله فهم بنية .net أكثر من دعمك لتطور تطبيقات بسرعة دون أن تفهم بنيتها الداخلية وكيفية تعاملها مع مكونات اللغة المختلفة .

وبالرغم من ذلك ستجد بين الفترة والأخرى تطبيق عملي نقوم بعمله سوية لتطبيق بعض المفاهيم التي تعلمناها لربط التعليم النظري بالتطبيق العملي لتحصيل أكبر كم من الفائدة من هذا الكتاب ... لذا ستجد بين طيات الأكواد الموجودة في هذا الكتاب عدد كبير من الأمثلة والتطبيقات الصغيرة لكن لا تنتظر مني أن أقول لك جرب هذا التطبيق مثلاً خطوة بخطوة .

أسأل الله أن يكون هذا الكتاب مفيداً، فما كان فيه من صواب فمن الله، وما كان من خطأ فمن نفسي والشيطان، والله الهادي إلى سواء السبيل.

أحمد جمال خليفة

نبـذة عن المؤلـف

أحمد جمال خليفة عبد العال .

خريج كلية الحاسبات والمعلومات – حلوان – 2007

جمهورية مصر العربية – القاهرة .

البريد الإلكتروني : A-Gamal@windowslive.com

الموبايل : 0020108011792

السيرة الذاتية بالكامل تجدها هنا http://hammada2091.googlepages.com/cv.pdf

مشرف عام منتديات فيجوال بيسك للعرب

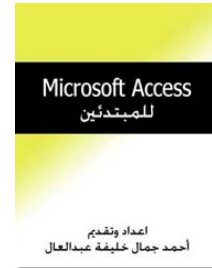
www.vb4arab.com

مدونة شخصية

www.AhmedGamal-Space.blogspot.com

مدونة تقنية

www.AhmedGamal-Technical.blogspot.com

كتب سابقة

شـكـر و عـرفـان

لا يشكر الله من لا يشكر الناس ، ومؤلف هذا الكتاب لديه في الواقع قائمة طويلة بأشخاص كثيرين يستحقون الكثير من الثناء والدعاء ، لذا اسمح لي أن أسرد أسماءهم هنا ، حيث أهدف من ذلك لئن أتذكر دوماً أن هذا الكتاب لم يكن ليكون كذلك لولا فضل الله علي أولاً وآخراً بالتعرف والاستفادة من هؤلاء الأشخاص حتى جاءت الثمرة المتواضعة التي تراها بين يديك الآن ، لذا أقدم شكراً خاصاً للأستاذ تركي العسيري صاحب كتابي (نحو برمجة كائنية التوجه) ثم (برمجة إطار عمل شيئي من خلال .net) حيث كان لكتبه كبير الأثر في توجيه مساري في عالم البرمجة ، زاد على ذلك أن شرفني بتقديمه لهذا الكتاب بكلمة أعطاني فيها أكثر من حجمي كثيراً ، لذا أرجو من الله أن أكون عند حسن ظن أستاذي بي ، وألا أخيب رجائه فيما يتوقعه مني .

الشكر أيضاً موصول للمهندس محمد سامر سلو حيث استفدت كثيراً من ومقالاته في تجهيز هذا الكتاب .

كما أجدد الشكر لإدارة منتدى فيجوال بيسك للعرب ومشرفيه على تشجيعهم لي لتقديم كتاب في هذا المجال ، أخص بالذكر منهم الأستاذ عبد الله العتيق مدير الموقع والأخ أحمد بدر ، وباقي الإخوة الكرام ...

الشكر أيضاً لكل من المهندس أحمد عيسوي، المهندس محمد النبهاني على مساعدتهم لي لتنقيح هذا الكتاب ، فالنسخ الأولى من الكتاب كانت تزخر – ولا تزال - بالأخطاء ، أشكر لهم تفريغ جزء من وقتهم لقراءة الكتاب وتوجيهي ومساعدتي على تدارك هذه المشكل قبل إخراج النسخة الأخيرة .

شكر خاص جداً ...

شكر خاص جداً للمهندس الطبيب وليد بوالطمين ، والمهندسة نورهان عادل ليس فقط لمساعدتهم في تنقيح محتويات الكتاب ، لكن لئن الكتاب الذي تراه أمامك هم من قاموا باخراجه ليكون بهذا الشكل بدءاً من تصميم الصفحات وأسلوب العرض والكتابة والأمثلة والتنسيق ، وانتهاء بتصميم الغلاف ، أكرر شكري لهم وأقدر لهم كثيراً ما بذلوه من جهد لإخراج هذا الكتاب في أبهى حلة .

الشكر عام أيضاً لكل من استفدت منهم في إعداد وإنتاج هذا الكتاب ، واعتذر لمن لم خانتني الذاكرة فلم أذكره هنا ، الشكر أيضا لجميع من استفدت منهم في حياتي العامة أو في دراستي ، لأهلي بالطبع ولجميع أصدقائي بلا استثناء ، وقبل ذلك الحمد لله أولاً وآخراً على توفيقه وإعانته لي لإتمام هذا الكتاب .

هذا الكتاب

- أسماء البرامج المذكورة في هذا الكتاب مثل Windows ،Office وغيرها هي علامات تجارية مسجلة لأصحابها و الكاتب يحق بملكيتها لأصحابها وان لم يشر إلى ذلك مباشرة للاختصار.

- تم اختبار اغلب محتويات المادة العلمية للملخص ، إلا أن الكاتب غير مسؤول بأي حال عن محتوياتها أو سوء استخدامها.

- حقوق الكتاب محفوظة للمؤلف ، ولا يحق طباعته أو توزيعه أو استخدامه لأي غرض تجاري بدون أذن من المؤلف.

- أغلب الأكواد تم تجريبها داخل الكتاب ، ولكن هذا لا يمنع وجود أكواد غير مجربة أو مجرب اتجاه واحد منها فقط #C أو VB.net ، لذا سأكون سعيداً لو ارسلت نتائج تجاربك على بريدي الإلكتروني.

- وبنفس الطريقة لو وجدت خطئاً إملائياً أو تعبيرياً سأكون أسعد لو راسلتني لتخبرني به.

- بعض الأكواد الموجودة ليست من تأليف الكاتب ، بل منها منقول وتمت الإشارة إلى عمليات النقل حال وجودها ، ومع ذلك فالمؤلف يعتذر عن أي نقل غير واضح أو لم تتم الإشارة لمصدره بصورة صحيحة.

- روابط المواقع الموجودة في الكتاب تم التحقق من سلامتها وقت إنتاج هذا الكتاب إلا أن الكاتب لا يضمن عملها أثناء قراءتك لمحتوياته.

إنتاج و تصميم موقع فيجوال بيسك للعرب vb4arab.com

محتـويات الكتـاب

يضم هذا الكتاب مجموعة من أساسيات عالم net. وتحديداً net 2008. عبر عدد من الفصول والأبواب ، هذا موجزها:

الباب الأول: هذا الباب ليس أكثر من كونه مقدمة عن محتويات الكتاب والفهرس وبيانات عن كاتب الكتاب والاتفاقية.

الباب الثاني: هنا تبدأ الإنطلاق في عالم net 2008. حيث يقودك هذا الجزء لتحميل نسختك من الفيجوال ستوديو وبرمجة برنامجك الأول وكيفية الترقية بين الإصدارات المختلفة.

الباب الثالث: هذا الباب مخصص لتحفيزك على الانطلاق في عالم . net، حيث يسرد لك بصورة مختصرة نبذة عن التقنيات الجديدة التي ربما تقنعك بضرورة اقتحام عالم . net، إذا لم تكن مبرمجاً من قبل فهذا الباب ليس مخصصاً لك سوى لاستعراض الجديد فقط.

الباب الرابع: هنا تجد وصفاً لمفاهيم بيئة.net الرئيسية وعناصرها المختلفة.

الباب الخامس: من هنا تستطيع الانطلاق في تصميم برامجك، حيث يعرض لك هذا الباب كيفية البدء بالبرمجة ومن ثم كيفية الانطلاق في عالم Visual Studio كبيئة البرمجة الخاصة بك.

الباب السادس: هنا سنبدأ باستعراض عناصر اللغة الأساسية وكيفية كتابتها، هذا الباب موجه بصورة أساسية للمبتدئين وهو فقط تذكير بأساليب كتابة اللغة للمحترفين.

الباب السابع: استمرار للباب السابق ولكن بصورة متقدمة نسبياً.

الباب الثامن: هذه هو مدخلك للبرمجة كائنية التوجه OOP حيث يمكنك التعرف على الفئات ومحتوياتها وكيفية العمل من خلالها .

الباب التاسع: الواجهات Interfaces وخصائصها واستخداماتها في البرمجة.

الباب العاشر: هذا الباب لك من أجل اقتناص أخطاءك في الكود وكيفية معالجتها وتنقيحها.

الباب الحادي عشر: يسرد هذا الباب بالتفصيل أنواع ال Collections واستخداماتها المتكررة في برامجنا المختلفة .

الباب الثاني عشر: البرمجة المتقدمة هذه المرة ، حيث ستتعرف على عناصر أكثر تقدماً في لغة البرمجة وفي عالم .net

الباب الثالث عشر: طرق الإدخال والإخراج المختلفة وكيفية التعامل مع الملفات .

الباب الرابع عشر: الانطلاق في عالم برمجة الWindows Forms والأدوات التي تحتويها والخصائص والرسم وخلافه .

الباب الخامس عشر: يأخذك في رحلة سريعة في عالم الرسوميات ثنائية الأبعاد من خلال GDI+ .

الباب السادس عشر: بداية لتقنية WPF المختصة بالرسوميات ثنائية وثلاثية الأبعاد وأدواتها وبرامجها المختلفة .

الباب السابع عشر: في هذا الباب يمكنك الانطلاق في عالم برمجة قواعد البيانات من خلال ADO.net وما يستلزم ذلك من قواعد البيانات SQL Server وخلافه .

الباب الثامن عشر: التقنية الجديدة من مايكروسوفت للاستعلام LINQ يمكنك البدء بها هنا.

الباب التاسع عشر: هنا نبدأ في عالم تقنية WWF الجديدة من مايكروسوفت لإدارة المشاريع.

الباب العشرون: أيضاً مع عالم WCF لمشاركة البرامج والتطبيقات من مايكروسوفت.

الباب الحادي والعشرون : مجموعة من المواضيع الهامة تتعلق بتشغيل برنامجك على الأجهزة المختلفة قبل الانطلاق في عالم الويب في الباب اللاحق .

الباب الثاني والعشرون : الباب الأخير ، يمكنك هذا الباب من تطوير تطبيقات ويب تفاعلية من خلال ASP.net .

أرجو من الله أن يكون في بعض محتوياتها الفائدة والنفع...

لمن هذا الكتاب؟

إجابة هذا السؤال هي الأصعب بالنسبة لي طوال مشواري في عمل هذا الكتاب، فهذا الكتاب لا يستهدف مستوى معين وليس مخصصاً لفئة معينة، فهو يحاول أن يقتبس من كل بستان زهرة، ويضعك على أول الطريق أياً كان مستواك.

فإذا كنت مبتدءاً فأبواب الكتاب الأولى توضح لك ربما أساسيات البرمجة ، أما لو كنت محترفاً فيكفيك معرفة الجديد فقط والانطلاق في الأبواب التي تختارها لنفسك ، أما لو كنت متوسطاً فهذا الكتاب سيكون نقطة انطلاق جيدة لك في عدة فروع من عالم net 2008. .

لذا أرجو أن لا يمل المحترف من كثرة التكرارات في الكتاب ، أو توضيح الواضح خصوصاً في الأساسيات ، حيث يمكنه تجاوز النقاط الأساسية والدخول فوراً في النقاط التي يراها هامة بالنسبة له ، أما إذا كانت هذه هي المرة الأولى لك للانطلاق في عالم البرمجة فحاول ألا تفوت شيئاً من محتويات الكتاب ، الفصول الأولى بالنسبة لك هي أساس كل شيء بعد ذلك .

يهمني جداً أن تطلع على محتويات الكتاب في الصفحة السابقة ، فهي توضح لك بعض الأبواب التي سيكون من المفضل تجاوزها لك كمبتدئ ، أو كمحترف ايضاً . لذا فضلاً لو كنت تجاوزت الصفحة السابقة أن تعيد النظر فيها مرة أخرى.

التعريـــب

في كل الكتب التي قرأتها باللغة العربية – أو أغلبها – كنت أجد نوعاً مع عدم الراحة مع قراءة الترجمة ، وفي مثل هذه الحالات كنت أفضل الإطلاع على الكلمة الأصلية باللغة الإنجليزية، وهذا هو الأسلوب الذي حاولت قدر المستطاع انتهاجه ضمن هذا الكتاب .

هناك بعض الترجمات لا تبدو سيئة ، مثل الجمل الشرطية وحلقات التكرار كترجمة لكلمتي Conditions و Loops ، هناك ترجمات ربما تبدو أقل جودة ولكنها لا تعد سيئة مثل تعريف ال Class على أنه (فئة) وتعريف ال Interface على أنها (واجهة) ، المتغير بديلاً ل Variable وهكذا ، في هذه الحالة ستجد في العادة تم ذكر الترجمة مرة واحدة بديلاً عن ذكرها في كل مرة ، أما ترجمة مثل (المشيدات ، المهدمات ... الخ) وخلافه فلم اتركها كلها ولم استخدمها كلها، استخدمت فقط ما رأيته مناسباً إلى حد ما وتركت البقية.

وعلى كل ثق بأنك لن تجد كلمة عربية لم يتم ذكر الكلمة الأصلية لها في أي مكان ، أيضاً لا تحاول الاعتماد على ترجمتي في قراءة كتاب عربي آخر ، فحتى اللحظة لا يوجد إي نوع من الاتفاق بين الكتاب على ترجمة أغلب مصطلحات الكومبيوتر فضلاً عن أن نتحدث في اتفاق عن ترجمة مصطلحات البرمجة .

الباب

2

تحميل نسختك من فيجوال ستوديو 2008

.1 النسخة الكاملة من Visual Studio 2008

يوجد عدة نسخة متاحة للتحميل من Visual Studio 2008 على موقع مايكروسوفت، النسخة الكاملة Professional Edition غير متاحة بصورة مجانية ولكنها موجودة بصورة تجريبية لمدة 90 يوم فقط، تجدها على هذا الرابط:

رابط

http://msdn.microsoft.com/en-us/vs2008/products/cc268305.aspx

النسخة التي يتم تحميلها من هذا الرابط تكون على شكل ملف iso.* برنامج Power ISO هو برنامج يستخدم لمحاكاة وجود CD أو DVD في الملفات التي يتم تحميلها ولا بد من وجود القرص الخاص بها .

يمكنك تنزيل برنامج Power ISO من هنا:

رابط

http://www.freedownloadscenter.com/Utilities/Backup_and_Copy_Utilities/PowerISO.**html**

إذا كنت ترغب في الترقية للنسخة الكاملة ، قم بطلب ال Product Key من مايكروسوفت من هذا الرابط :

http://msdn.microsoft.com/en-us/vs2008/products/cc263903.aspx

ومن ثم قم باختيار Add or Remove Programs وقم باختيار Visual Studio 2008، ومن ثم قم باختيار Change/Remove ... قم بإدخال ال Product Key الذي قامت مايكروسوفت بإرساله لك.

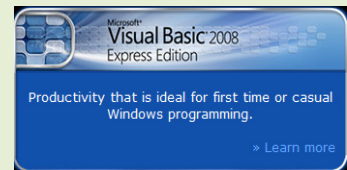
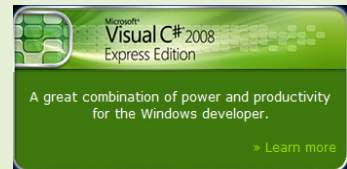
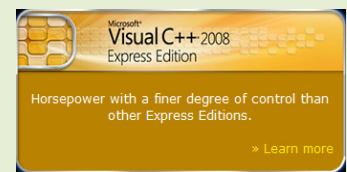
2. النسخ المجانية من Visual Studio 2008

إضافة للنسخ الكاملة التي توفرها مايكروسوفت، توفر أيضا حلول مجانية للمطورين، تجدها جميعاً على الرابط التالي:

رابط

http://www.microsoft.com/express/product/default.aspx

هذه النسخ ال Express اقل في الإمكانيات من النسخ الكاملة Professional Edition ولكنها تعد الحل الأكثر استخداماً من قبل مطوري الدوت نت ، تجد النسخ التالية في الرابط السابق :

النسخة	البرنامج
	نسخة من ال net 2008. بأسلوب كتابة ال Basic على بيئة تطوير net Framework. http://go.microsoft.com/?linkid=7653517
	نسخة من ال net 2008. بأسلوب كتابة ال C# على بيئة تطوير net Framework. http://go.microsoft.com/?linkid=7653518
	نسخة من ال net 2008. بأسلوب كتابة ال C++ على بيئة تطوير net Framework. تستخدم لإنتاج التطبيقات المختلفة http://go.microsoft.com/?linkid=7653520
	بيئة تطوير مبنية على net Framework. توفر لك أدوات متطورة لإنشاء تطبيقات ويب متقدمة http://go.microsoft.com/?linkid=7653519

	نظام قواعد البيانات المشهور، يوفر لك نسخة مجانية لإنشاء قواعد بيانات سهلة التعامل مع برامجك من Net http://www.microsoft.com/express/sql/default.aspx
	أحد أدوات مايكروسوفت الجديدة التي تسهل عملية بناء ال gadgets وال mash-ups http://www.microsoft.com/express/popfly/default.aspx

الجدول 2. 1. النسخ Express المكونة لبيئة التطوير net Framework.

3. ترقية مشروعك الذي يعمل على VS 2005 إلى VS 2008

يمكنك فتح نسخة من مشروعك المبني من خلال net 2005. ليعمل على net 2008. للاستفادة من خصائصه .

- قم بفتح المشروع من خلال VS 2008 وسيظهر لك معالج الترقية مباشرة .
- قم باختيار Next .
- قم باختيار عمل نسخة احتياطية من الملفات، قم باختيار المكان الذي تود حفظ النسخة القديمة احتياطياً فيه.
- قم باختيار Finish .
- سيتم إنهاء الترقية ، وستحصل على تقرير بالانتهاء بعد انتهاء العمل .
- حتى اللحظة فإن مشروعك يعمل من خلال 2008 ولكنه لا زال متوافقاً مع 2005، لتعديل الخصائص لتصبح متوافقة مع 2008 فقط قم بالخطوات التالية:
- من Solution Explorer قم باختيار خصائص المشروع.
- من التبويب Compile قم باختيار Option Infer إلى ON.
- ومن Advanced Compile Options قم باختيار Target framework ليصبح net. Framework 3.5 بدلاً من net Framework 2.0.

- الآن سيكون عليك إضافة بعض المراجع التي توجد تلقائياً مع 2008، في أي مشروع قم بمعرفة الخيارات التي تريدها، قم بالانتقال إلى References وقم بإضافة المراجع المطلوبة.

الباب

3

الجديد مع ال .net

2008

في هذا الجزء من الكتاب ستساعدني على افتراض نقطتين هامتين، النقطة الأولى هي انك تود البدء حقاً بتعلمnet 2008. ، النقطة الثانية أنك لم تبرمج للمرة الأولى ، سنتحدث في هذا الباب لبعض الوقت عن التقنيات الجديدة التي ظهرت في .net 2008 ، فلو لم تكن مبرمجاً في الأساس فلن يضيرك أن تقلب صفحات هذا الباب لتنطلق مباشرة للباب التالي .

المقصود بهذا الباب ليس فقط التقنيات العامة الجديدة التي ظهرت مع. net 2008 بل كل التقنيات التي أصبحت تلقى الدعم الكامل داخل إطار عمل .net Framework من تقنيات فقط والتي سيتم شرحها بدورها في دروس منفصلة لاحقاً ، أما الجديد في عالم البرمجة فسيتم التعرف عليه في دروس قادمة .

1. تقنية Silverlight

كجزء من سياسات مايكروسوفت الرامية إلى محاولة تملك أغلب المفاتيح الرئيسية في مجال التقنية والتكنولوجيا كانت تقنية Silverlight كمحاولة لزعزعة استقرار محركات Flash على عرش الويب كحل أمثل للتطبيقات التي تستخدم الملتيميديا المتحركة أو الاعتماد على طرق الإدخال والإخراج .

وربما بعد أن أحست شركة Adobe المالكة لمحرك Flash بالخطر - أو ربما في سياق تطويرها لمحركها، من يدري ! - بدأت تحركها في التطوير السريع لمحرك فلاش الذي لم يشهد تغييرات جذرية منذ اشترت شركة Adobe حقوقه من شركة Macromedia ، فأصبح لشركة Adobe الآن ثلاث محركات تعتمد على Action Script ، أولها وأشهرها هو Flash ومن ثم تقنية Flex وأخيراً تقنية Apollo ، فيما دخلت شركة Sun على خط المنافسة من خلال منتجها JavaFX Script وهذا بالتأكيد سينصب في مصلحتنا في النهاية ، وربما تكون النقطة يتميز ال Silverlight عن باقي محركات الفلاش بدعمه لتشغيل فيديو عالي الدقة HD Video والتي على حد علمي لم تحاول شركة منهم دخول هذا المجال بعد - على حسب معلوماتي - .

لتتعرف عن المزيد عن هذه التقنية برجاء زيارة موقع Silverlight :

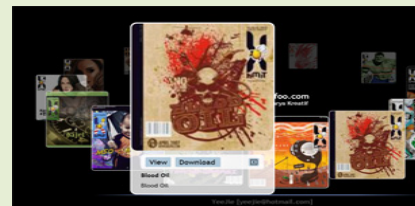
رابط

http://www.microsoft.com/silverlight/default_ns.aspx

بداية قم بتحميل Silverlight من الرابط الموضوع Get Silverlight، لتبدأ بعدها بتجربة امكانيات وقدرات Silverlight .

قم بتحميل البرنامج وعمل Setup له ، ومن ثم قم بتجربة الموقع مرة أخرى ، واستمتع بإمكانيات Silverlight ... ولاحظ الفرق .

جرب المواقع التالية والتي تعتمد ايضاً على تقنية Silverlight :

صورة	الموقع
	http://silverlight.net/samples/1.0/Page-Turn/default.html
	http://www.yeejie.com/silverlight_magazine/

الجدول 3. 1. بعض المواقع التي تعتمد على تقنية ال Silverlight.

2. تقنية LINQ

تقنية جديدة من مايكروسوفت في مجال قواعد البيانات ، حيث تتيح لك هذه التقنية تحويل الاستعلام إلى جزء من لغة البرمجة بما يتيح لك التعامل مع الجداول والحقول والكائنات بصورة مباشرة من خلال لغة البرمجة .
كمثال سريع يمكن افتراض جملة الاستعلام هذه:

كود	SQL

```sql
Select companyname from customers
```

يتم تنفيذها باستخدام أي طريقة ... ليس هذا ما يعنينا ، ولكن تجد ناتج الاستعلام في RS.Fields("FieldName") مثلاً حسب طريقة القراءة ، حيث تستطيع قراءتها ، لعمل استعلام آخر يمكنك القراءة من جديد ، يمكنك عمل Loop عادية للقراءة ومن ثم البحث في الكائن بالطريقة العادية:

كود	VB.NET

```vb
Do While (Rs.Read())
    Console.WriteLine(Rs.Fields("CompanyName"))
Loop
```

كود	C#

```csharp
do
{
    Console.WriteLine( Rs.Fields("CompanyName"));
}
while(Rs.Read());
```

كان هذا هو الاختيار المتاح لك للتعامل مع قواعد البيانات ، أما الآن فقواعد البيانات يتم التعامل معها على شكل Classes لها Members و Methods وFunctions ولها Operators خاصة بها ، لنفترض هذا الأمر من داخل ال net. مباشرة.

كود	VB.NET

```
Dim queryResults = From cust In customers _
                   Select cust.CompanyName
```

كود	C#

```
var queryResults = from cust in customers
                   select cust.CompanyName;
```

والآن أصبح بإمكانك التعامل مع queryResults كفئة مباشرة لها خصائصها وحقولها التي أصبح بإمكانك قراءتها مباشرة ، سنتعرف في باب لاحق عن هذه التقنية بالتفصيل .

3. تقنية WPF

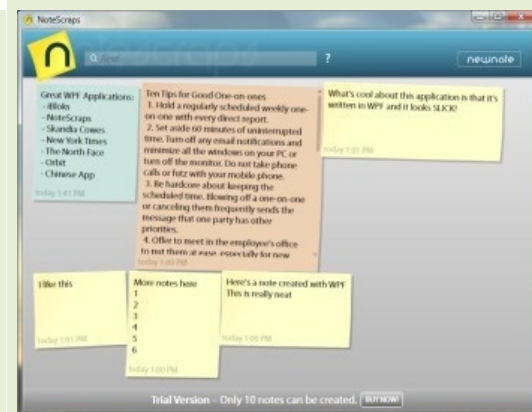
واحدة من التقنيات الجديدة التي شهد WinFx ظهورها إلى النور وهي اختصار ل Windows Presentation Foundation تعتمد على تقنية أخرى هي XAML وهي لغة تستخدم لوصف الواجهات ثنائية أو حتى ثلاثية الأبعاد، أثناء عملك على WPF من خلال net 2008. يتم توليد كود XAML بصورة فورية ،فيما يظل بإمكانك ايضاً الكتابة ب XAML إن أردت . يتم الفصل الكامل بين ال XAML والكود العادي كما كان يحدث في صفحات الويب .

لماذا استخدام WPF ؟

الفصل الكامل بين لغة البرمجة وبين التصميم باستخدام XAML قادنا إلى ظهور حزمة من برامج التصميم المتخصصة التي تولد كود XAML ، تخيل نفسك تصمم برنامجك على فوتوشوب ، أو على فلاش وتكتب الكود في net. ، أليس هذا سيمنحك مزيداً من التحكم وقوة التصميم التي لم تكن لتتاح لك لو كنت أكملت التصميم على Visual Studio المصمم أصلا لخدمة الكود وليس التصميم ، ايضاً ستجد مجموعة من الحزم الجاهزة التي ستساعدك على تجميل وتحسين مظهر البرنامج.

قبل النهاية، إذا كنت متابعاً معنا فأنت بالتأكيد قمت بتحميل net 2008. أو على الأقل net Framework 3.5.، لذا فأنت مؤهل للاطلاع على هذه الأمثلة، وشاهد الفرق بينها وبين

الواجهات التقليدية ، ربما لا تحتاج إلى تحميل البرنامج فقط اطلع على الفيديوهات الموجودة أو نماذج الصور ، وكلي ثقة انك ستغير مفهومك حول جمال الواجهات إلى الأبد :

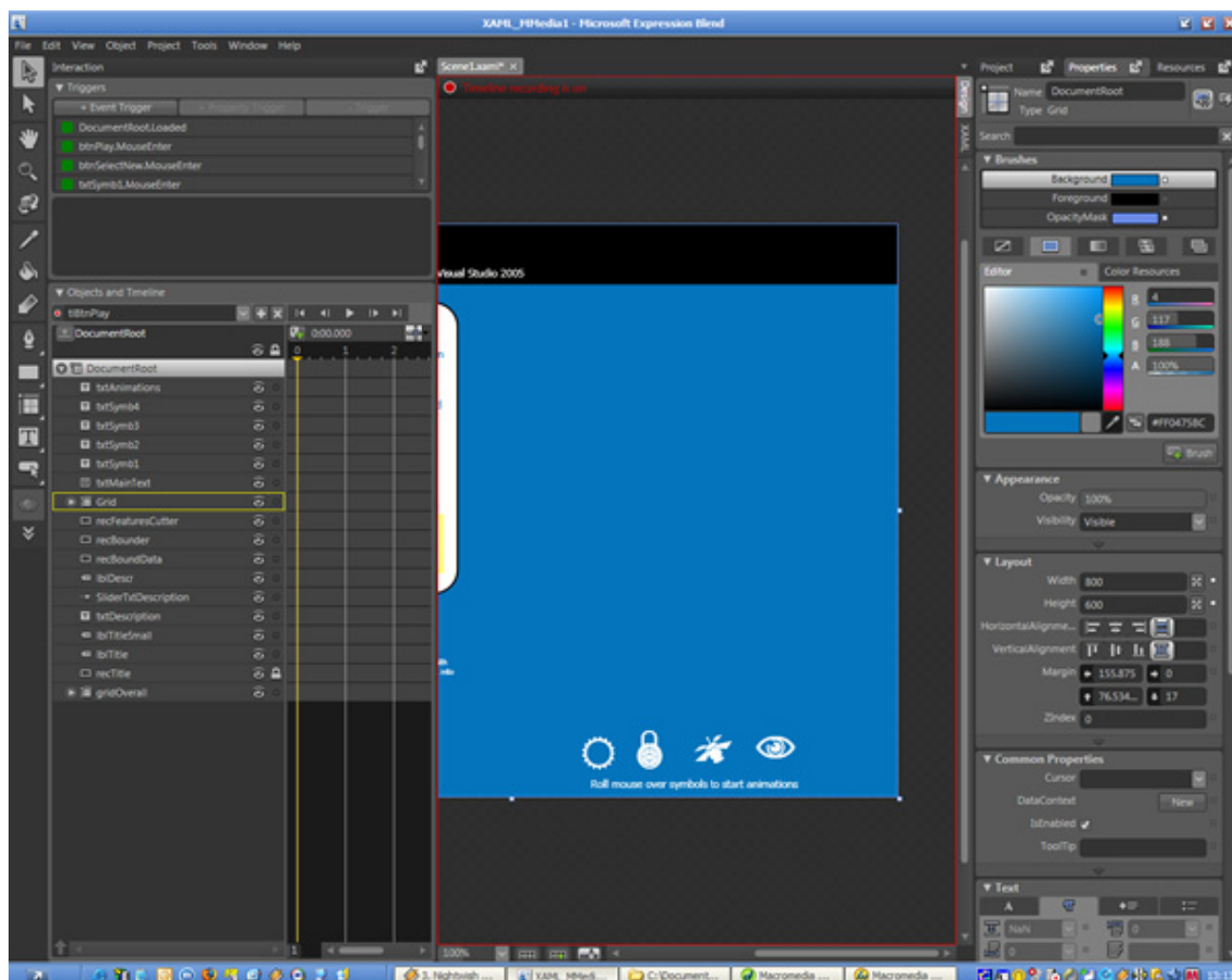
صورة	الموقع
	http://blogs.msdn.com/tims/archive/2007/02/02/great-wpf-applications-6-fnac-com.aspx
	http://blogs.msdn.com/tims/archive/2007/03/05/great-wpf-applications-12-roxio-central.aspx
	http://blogs.msdn.com/tims/archive/2007/02/09/great-wpf-applications-8-notescraps.aspx

الجدول 3 .2. بعض الفيديوهات لتقنية ال WPF.

4. مجموعة Expression

لنعد إلى الوراء قليلاً في عملية التصميم ، حيث يقوم المصمم برسم الواجهة على الفوتوشوب أو الفلاش أو غيره من برامج التصميم ، يتم لاحقاً تقطيع الصور ووضعها في الصفحة، أو برمجة بعض الأوامر والحركات من خلال Flash ومن ثم نقلها لموقعك أو لبرنامجك ... وهكذا ، تجد فصلاً تاماً بين عملية التصميم وبالبرمجة.

أما الآن ومع ستوديو Expression كاستديو اعلنت مايكروسوفت عن تطويره لاحتياجات مصممي التطبيقات ، حيث أصبح الآن بإمكانك استخدام برنامج تصميم من انتاج مايكروسوفت يقوم بإنتاج كود XAML يمكنك استخدامه في تطبيقاتك مباشرة ، كما يمكنك البرمجة من خلال Expression Blend أيضاً.

يصدر MS Expression مع أربع تطبيقات رئيسية ، Blend لتطوير الملتيميديا والصور والمؤثرات الحركية - شديد الشبه ببرنامج فلاش - ، التطبيق الثاني هو Design وهو موجه بالأساس لخدمة تطبيقات الويندوز - شديد الشبه بفوتوشوب - ، الثالث هو لأجل ال WEB حيث أصبح تصميم مواقع الإنترنت بمستوى عالي جداً ، وهو التطور ل Microsoft FrontPage ولكن بإمكانيات متقدمة جداً من اجل دعم AJAX و XAML ،الأخير مخصص للتعامل مع الميديا والفيديو وخلافه باسم Expression Media .

التطبيق الأكثر شهرة بينهم هو Blend حيث أصبح تحويل موقعك أو برنامجك إلى موقع غني بالحركة وبالمؤثرات وخلافه امراً في غاية السهولة ، هذه هي الشاشة الرئيسية للبرنامج:

الصورة 3. 1. استديو ال Expression Blend

يمكنك تحميل نسخة تجريبية لمدة 21 يوم من موقع مايكروسوفت الرسمي - أو من موقع التقنية - حيث قامت مايكروسوفت بشراء التقنية من إحدى الشركات ال Partners لمايكروسوفت كما سنتعرف على هذه التقنية بصورة مفصلة ضمن دروس لاحقة في هذا الكتاب.

5. تقنية WCF

هي إحدى تقنيات مايكروسوفت الجديدة التي تخدم البرامج التي تعمل في وضع اتصال ، سواء على الشبكات أو برامج مثل الماسنجر وخدمات البريد الالكتروني وخلافه ... باختصار أي برنامج يتم فيه ربط جهازين ببعضهما فهذه التقنية موجهة لهذا الغرض.
بعد Winsock في فيجوال بيسك 6 وفئة Sockets مع net. ، جاءت لنا مايكروسوفت بتقنية جديدة من اجل عمليات الاتصال تحت اسم WCF وهي اختصار ل Windows Communication Foundation.

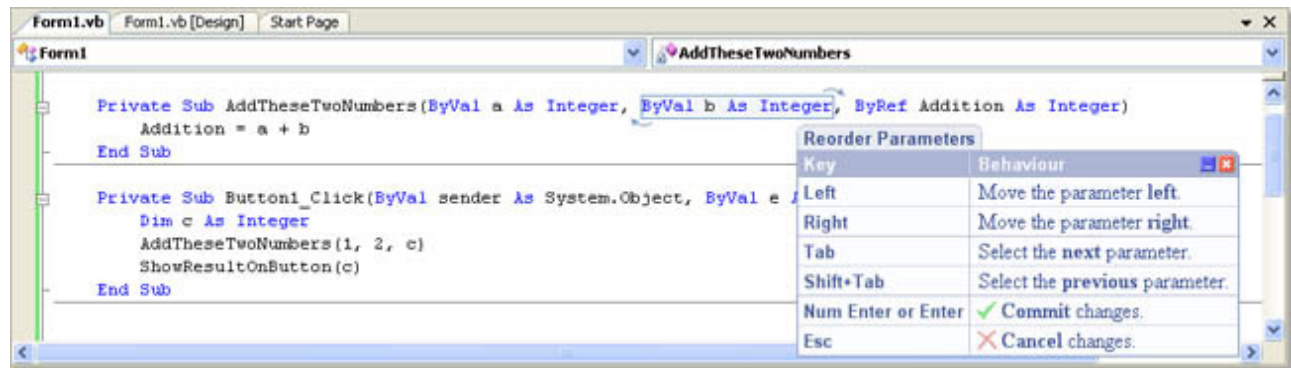
أما لماذا هذه التقنية ، فباختصار لأنك ستستطيع عمل تطبيق عميل وخادم Server And Client آمن ومتكامل من خلال اقل من عشرة اسطر فقط من الكود!!!
كما أنها أسرع من التقنيات السابقة، في درسنا المفصل عن WCF سوف نتطرق للمقارنات بينها وبين باقي التقنيات.

6. Code Refactoring

إحدى المميزات الجديدة التي جاء بها ال IDE الخاص بالفيجوال ستوديو ، الفكرة باختصار هي مجموعة من الوسائل المساعدة على تنظيم الكود وتسهيل التعامل معه.
في العادة يتم توفير مثل هذه البرامج على شكل برامج مساعدة، ومع net 2008.أصبح واحد منها مضمن بصورة افتراضية ، تمكنك هذه الإضافة من إنشاء خصائص لمتغير بضغطة زر ، لدمج المتغيرات في فئات ، التعديل والتغيير في الفئات والبارميترات وخلافه كل هذا بواجهة مرئية.
ايضاً هناك برامج أخرى مثل هذا البرنامج لأجل فيجوال بيسك Refactor! for Visual Basic 2008. البرنامج موجود على هذا الرابط:

رابط

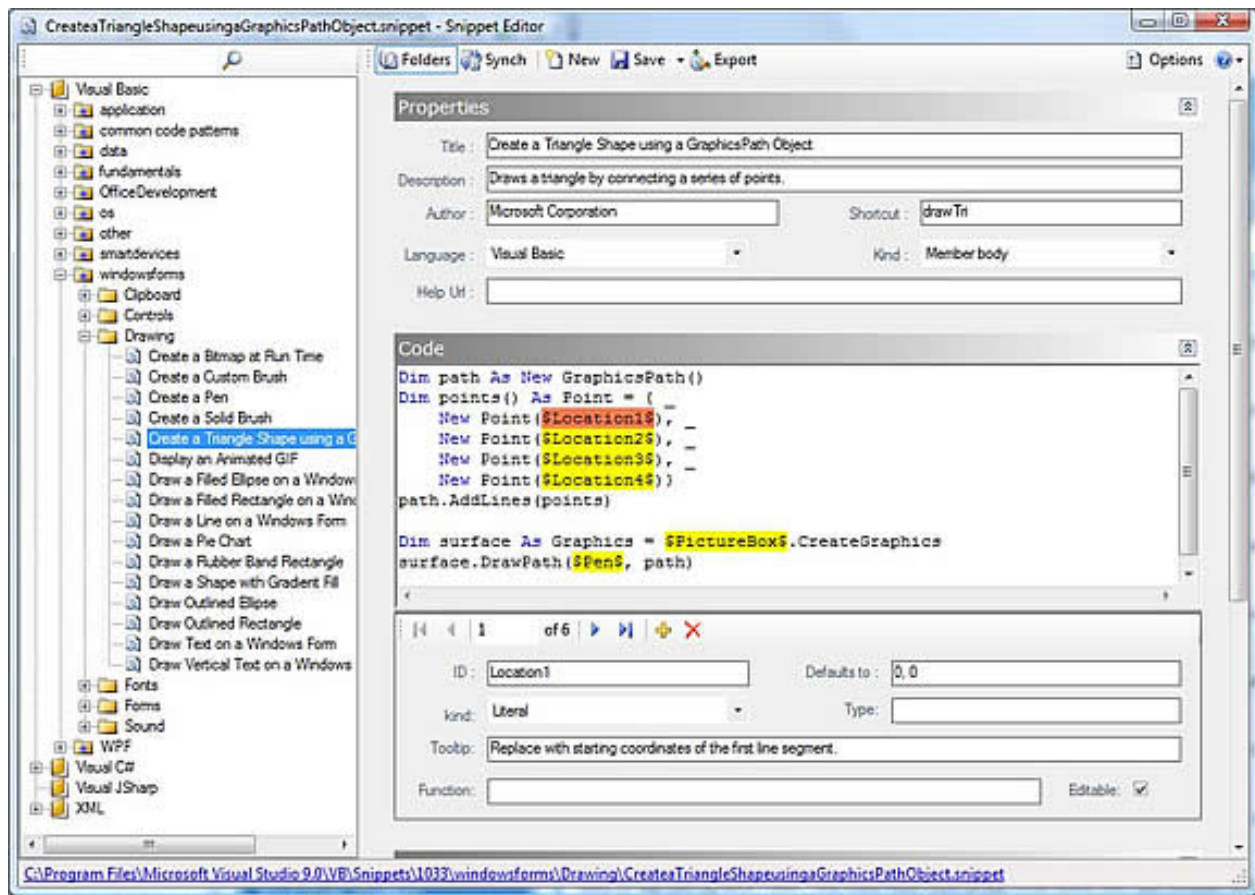
http://msdn.microsoft.com/en-us/vbasic/bb693327.aspx

وهذه صورة لبعض التعديلات التي يضيفها على نافذة كتابة الكود:

الصورة 3. 2. Refactor! For Visual Basic 2008

توفر مايكروسوفت ايضاً خاصية Code Snippet وهي مضمنة بصورة افتراضية في 2008 ، تمكنك هذه الخاصية من إدراج كود كامل لعملية معينة بدون كتابة سطر واحد ، يتم ذلك باختيار المهمة المطلوبة من قائمة يمكن التحديث فيها واضافة اكواد جديدة ، وكالعادة هذا البرنامج من اجل فيجوال بيسك وستجد مثيلاً له من اجل سي شارب:

رابط

http://msdn2.microsoft.com/en-us/vbasic/bb973770.aspx

وهذه صورة للبرنامج:

الصورة 3. 3. Code Snippet Editor

اخيراً وليس آخراً اذكر بأن هناك اصدارات تعمل على 2005 فليس الامر خاصاً بـ 2008 فقط ، ولكنها خاصية اصبحت الآن مضمنة افتراضياً مع Visual Studio 2008.

الباب

4

مفاهيم بيئة .net الرئيسية

1. لغات ال net.

لا تعتبر net. لغة واحدة ، بل هي عبارة عن معيار قياسي تستخدمه عدة لغات برمجة ... تم طرح هذا المعيار من قبل مايكروسوفت حوالي العام 2000 وتبنت العديد من الشركات هذا المعيار ليكون قاعدة انطلاقها في لغات البرمجة ، حيث تتميز جميع هذه اللغات بإن لها الكود المصدري نفسه ، وتترجم نفسها مروراً بنفس المراحل ولا تختلف سوى في طريقة الصياغة ... سنقوم بالتعرف على المكونات الاساسية لمحرك اللغة Engine كاملة .. ولكن قبلاً لنتعرف على بعض اللغات التي تبنت معيار net.

فبعد اللغات التي تصدرها مايكروسوفت بنفسها والتي تضم C# ، VB.net، J#.net ، اضافة إلىC++/CLI وأخيراً ... JScript.net هناك اللغات الأخرى التي تدعم معيارية net. مثل باسكال ودلفي ، كوبول وحتى LISP.net لبرمجة تطبيقات الذكاء الإصطناعي.

لا ننسى ايضاً الإصدارات التي تعمل على بيئة Linux ، ف mono مثلاً يدعم معيارية net. تحت بيئة اللينكس .. لكن في النهاية فقد كان المنتج الاساسي لخدمة بيئة net. هو C# ، لتنضم إليه لاحقاً VB.net ربما لاغراض تسويقية وبدأ بعدها تتابع اللغات ، لا ننسى ان انضمام f# للمجموعة قد يجعله يوماً ما الاسلوب رقم 1 للبرمجة بتقنية net.

تستطيع في النهاية الاطلاع على مجمل اللغات التي تدعم هذه المعيارية مع بعض التفصيلات عنها هنا:

رابط

http://www.dotnetlanguages.net/DNL/Resources.aspx

اضافة

في كتاب :

Pro C# 2008 .NET 3.5 Platform -Exploring the .NET universe using curly brackets

يطرح الكاتب تساؤلاً ويجيب عليه، لماذا كل هذا العدد من لغات net. ما دام سيتحولوا جميعاً في النهاية إلى maneged code ويجيب على هذا التساؤل بعدة اسباب :

- ان العديد من المبرمجين حساسين جداً للغة التي يبرمجون بها ويحبونها ، فبعضهم يفضل اسلوب الكتابة ب ; والأقواس ... فيما يفضل البعض الآخر الاسلوب الأكثر قابلية للقراءة مثل Syntax لغات Visual Basic ، لا ننسى هنا الدور التسويقي ومحاولة جذب جميع المبرمجين للعمل مع net. حيث يمكن لمبرمجي C# و Basic و C++ و Fortran و Delphi العمل جميعاً تحت منصة واحدة ، وهذا ما يجعل انتقالهم بينها اسهل اضافة لتجميع المبرمجين تحت مظلة واحدة.

- مشاركة الفريق الواحد مبرمجين بلغات متعددة ، حيث يمكن ان تجد في الفريق الواحد مبرمجين يبرمجون بعدة لغات ويتم اخراج منتج واحد في النهاية بعد ان كان من المفضل ان يكون المبرمجين لنفس اللغة ، هذا يفتح الاختيارات كثيراً امام الشركات في اختيار المبرمجين.

- بعض اللغات تتميز بنقاط قوة تضطر الناس لاستخدامها ، فمثلاً هناك لغات مميزة في العمليات الرياضية والمعادلات مثل الفورتران ، الكوبول مميزة ايضاً في التعاملات المالية والعمليات الحسابية ... هكذا تستطيع الاستفادة من هذه المميزات وتدمجها جميعاً مع بعضها تحت بيئة ال . net .

2. العناصر الأساسية لبيئة .net

تقدم net. بين طياتها ثلاث عناصر رئيسية تتحكم في دورة حياة البرنامج المعتمد على بيئة.net، هذه التقنيات الثلاث يرمز لها اختصاراً بأسماء: CLR, CTS and CLS.

تمثل ال CLR ال Runtime Layer للبرنامج ، وهي اختصار لCommon Language Runtime، مهمته الاساسية هي التحكم في الأنواع والمكونات الرئيسية لل - net. المكتبات والدوال - وحجز المتغيرات في الذاكرة وتقسيم الذاكرة والتعامل معها ، ال Threads والتنفيذ اضافة لبعض اختصارات الأمان ،" باختصار شديد تشكل ال CLR المرحلة الاساسية لترجمة أوامر net. وتنفيذها على جهاز الكومبيوتر.

ال Common Type System والذي يتم اختصاره ب CTS مختص بالتعامل مع انواع البيانات المدعومة للعمل ضمن بيئة net.، وكيف يتم التعامل بينهم وبين بعضهم ، مع مراعاة وجود بعض الانواع غير المدعومة في جميع بيئات net. والتي توجد بها اختلافات بين بيئة وأخرى ، أما الأخيرة وهي Common Language Specification والتي يتم اختصارها بالرمز CLS فهي مختصة بالتعامل فقط مع انواع البيانات القياسية والمدعومة من جميع اصدارات بيئة Net . .

وبهذا نستطيع ان نقول ان البرنامج المصمم باستخدام C# مثلاً ويستخدم فقط الأنواع المتاحة في CLS هو متوافق تماماً للعمل على نفس توزيعة net. من اجل العمل على Linux المسماه mono، اما لو خرجت خارج نطاق الCLS إلى CTS فلن تضمن ان يتم تنفيذها بنفس الكفاءة في كل الأنظمة المختلفة.

3. ال Base Class Library

توفر تقنية net. ما يعرف باسم Base Class Library، وهي عبارة عن مجموعة من الفئات Classes تحتوي على الأوامر والدوال الرئيسية في بيئة. net، هذه الفئات موجود في جميع بيئات net. ويتم التعامل معها ومع خصائصها بنفس الطريقة ، و تضم العناصر الاساسية مثل التعامل مع الملفات وقواعد البيانات ، الThreads، ال XML ، ال GUI وخلافه. وتساهم هذه المكتبة في جعل اسلوب البرمجة موحد وأسهل. وقابل للتكامل مع أي من اللغات التي تدعم Net.

هذا الجدول لبعض محتويات ال BCL من مايكروسوفت:

Namespace	Description
System	This namespace includes all the essential support you need for your programming, including base types (String, Int32, DateTime, Boolean, etc.), essential environmental support, and math functions, to name a few
System.CodeDom	all the support necessary to be able to create code, and run it, on the fly
System.Collections	The System.Collections namespace contains interfaces and classes that define various containers, such as lists, queues, bit arrays, hashtables and dictionaries.
System.Diagnostics	All the classes you need to diagnose your application, including event logging, performance counters, tracing, and process management APIs.
System.Globalization	This namespace includes fundamental support for Globalization, used throughout the rest of the Framework
System.IO	Includes fundamental Stream support which can be used by anyone, and then specifically targets the FileSystem (via File and Directory manipulation classes), SerialPorts, and Decompression
System.Resources	Used to allow an application to be translated into multiple languages, and then display the appropriate text based upon the current users language selection
System.Text	This namespace includes support for encodings, and Stringbuilder
System.Text.RegularExpressions	This namespace includes regular expression support, for robust parsing and matching of string data

الصورة 4. 1. بعض مكونات ال BCL - من ال MSDN -

وكترجمة سريعة ، يحتوي System على كل ما تحتاجه لتطوير تطبيقات net. ، يحتوي ايضاً على ال Data types والدوال الاساسية للعمليات الحسابية .. الخ ، باختصار شديد تجد أن System هي مجال الأسماء الرئيسي والتي تحتوي تحتها على باقي الفئات.

مجال الأسماء	الوصف
System.CodeDom	خاصة بكتابة الاكواد وتنفيذها
System.Collections	تحتوي على عدد من الانواع مثل Lists, Stack.
System.Diagnostics	يضم ال Events وال Counters والProcess
System.Globalization	يتعلق بتشغيل برنامجك على نظم مختلفة من حيث اللغات والاعدادات الاقليمية وخلافه.
System.IO	كل ما يتعلق بعرض واستقبال البيانات سواء عن طريق الملفات والمجلدات ، او عن طريق منافذ الكومبيوتر مثل الSerial Port
System.Resources	ال Resources التي تصف البرنامج ، يتيح لك عمل تطبيقات متعددة اللغات على سبيل المثال
System.Text	كل ما يتعلق بالتعامل مع النصوص
System.Text.RegularExpressions	التعامل مع ال Regular Expressions

الجدول 4. 1. بعض مكونات مجال الأسماء System

4 . net Assemblies.

في النهاية ومهما كانت لغة البرمجة وال Compiler الذي تستخدمه لبناء تطبيقات net.، فإن الكود يتحول في النهاية ل intermediate language وهو ما يعرف اختصاراً IL بالاضافة إلى بعض ال metadata ، ويتم وضعه في ملف اسمبلي قابل للعمل مباشرة سواء على شكل exe أو على شكل dll

سابقاً كان يرمز لل IL باسم MSIL وال MS اختصار لمايكروسوفت ، اما الآن فتمت اضافة اللاحقة C كاختصار ل Common بدلاً من MS السابقة.

يمثل ال IL أو ال MSIL المهام التي يقوم بها البرنامج والأكواد وما شابه ، اما ال metadata فتحتوي على وصف لجميع الأنواع والفئات التي استخدمتها في برنامجك ، ملف الاسمبلي نفسه الذي يضم ال IL وال metadata يتم وصفه ب metadata أيضاً ، يتم اضافة manifest ايضاً لهذه المجموعة ، هناك حالات تجد برنامجك فيها مرتبطاً باكثر من ملف اسمبلي ، وفي هذه الحالة لن يكون لديك سوى manifest واحد فقط في واحد منها يقوم بعمل البداية وربط الملفات مع بعضها البعض.

5. ال CIL

لنفترض هذا البرنامج بلغة C# من كتاب Pro CSharp 2008

كود	C#

```
// Calc.cs
using System;
namespace CalculatorExample
{
    // This class contains the app's entry point.
    class Program
    {
        static void Main()
        {
            Calc c = new Calc();
            int ans = c.Add(10, 84);
            Console.WriteLine("10 + 84 is {0}.", ans);
            // Wait for user to press the Enter key before shutting down.
            Console.ReadLine();
        }
    }
    // The C# calculator.
    class Calc
    {
        public int Add(int x, int y)
        { return x + y; }
    }
}
```

لو قمت بفتح الاسمبلي الناتج عن هذا الكود باستخدام أي تطبيق مناسب مثل ILDASM ستجد الكود

كود	CIL

```
.method public hidebysig instance int32 Add(int32 x,int32 y) cil managed
{
    // Code size 9 (0x9)
    .maxstack 2
    .locals init (int32 V_0)
    IL_0000: nop
    IL_0001: ldarg.1
    IL_0002: ldarg.2
    IL_0003: add
    IL_0004: stloc.0
    IL_0005: br.s IL_0007
    IL_0007: ldloc.0
    IL_0008: ret
}// end of method Calc::Add
```

حتى هذه المرحلة ، لم يتم تحويل الكود إلى Platform-Specific Instructions، يتم ذلك في المرحلة التالية من خلال ما يعرف باسم Jitter والذي يقوم بترجمة الكود الناتج إلى كود مناسب لامكانيات الجهاز ونظام التشغيل الذي يعمل عليه البرنامج.

بامكانك التعرف على المزيد عن هذا الموضوع من هذا الرابط من مايكروسوفت :

رابط

http://msdn.microsoft.com/en-us/library/f7dy01k1(VS.80).aspx

6. ال MetaData

مثال على MetaData التي تستخدم لوصف كل محتويات الكود من دوال وفئات وواجهات وخلافه

كود	Assembly Metadata

```
TypeDef #2 (02000003)
-------------------------------------------------------
TypDefName: CalculatorExample.Calc (02000003)
Flags : [NotPublic] [AutoLayout] [Class]
[AnsiClass] [BeforeFieldInit] (00100001)
Extends : 01000001 [TypeRef] System.Object
Method #1 (06000003)
-------------------------------------------------------
MethodName: Add (06000003)
Flags : [Public] [HideBySig] [ReuseSlot] (00000086)
RVA : 0x00002090
ImplFlags : [IL] [Managed] (00000000)
CallCnvntn: [DEFAULT]
hasThis
ReturnType: I42 Arguments
Argument #1: I4
Argument #2: I42 Parameters
(1) ParamToken : (08000001) Name : x flags: [none] (00000000)
(2) ParamToken : (08000002) Name : y flags: [none] (00000000)
```

يقوم الجزء السابق بوصف الكود الذي قمنا بكتابته في أول مثال ، يمكنك وانت تعمل على net. ان تقوم بالاطلاع على هذا الكود ، اغلب النقاط واضحة وتشرح نفسها ، اعتقد انه ربما لن تحتاج يوماً للتعديل اليدوي على هذا الملف.

7. ال Manifest

نسخة واحدة فقط من هذا الملف لكل برنامج حتى لو كان هناك اكثر من Assembly File، يصف هذا الملف الاسمبلي نفسه من حيث رقم نسخة الاسمبلي وملف الاسمبلي الرئيسي في حالة وجود اكثر من ملف ، هذا مثال على كود من هذه النوعية:

كود — Assembly Manifest

```
.assembly extern mscorlib
{
.publickeytoken = (B7 7A 5C 56 19 34 E0 89 )
.ver 2:0:0:0
}
.assembly Calc
{
.hash algorithm 0x00008004
.ver 0:0:0:0
}
.module Calc.exe
.imagebase 0x00400000
.subsystem 0x00000003
.file alignment 512
.corflags 0x00000001
```

8. Common Type System

سنركز في هذا الموضوع المختصر على CTS حيث سبق وذكرنا انه مختص بوصف الانواع المدعومة في بيئة net. وكيفية تعاملها مع بعضها البعض ، هذه الأنواع هي:

- Class
- Interface

- Structure
- Enumeration
- Delegate

سنتعرف على طريقة تعريف كل منهم في مرحلة مختلفة من الكتاب ، إلا ان ما يتعلق بنا الآن خلال الدرس الخاص بنا هو كيفية تعامل ال CTS معهم ، إذا كنت مبرمج تطبيقات عادية فلن تحتاج لمعرفة اكثر من فائدة ال CTS اما مبرمجو الأدوات او الذين يقومون ببناء Compiler او لغة برمجة ضمن بيئة تطوير net. فيحتاجون للتعرف على الخصائص الاساسية ، سأذكر مثالاً مختصراً على Class

هناك ما يعرف باسم Abstract Class ، ال Abstract Class لا يحتوي على اي كود - شديد الشبه بال - interface بحيث يمكن بعد ذلك عمل وراثة له قبل البدء في استخدامه حيث انه لن يحتوي على سطر كود واحد ، هذا مثال لما يعرف باسم Abstract Class

كود	C#

```csharp
class abst_class
{
    int sum(int x, int y);
    string name;
}
```

كود	VB.NET

```vbnet
Class abst_class
    Private Function sum(ByVal x As Integer, ByVal y As Integer) As Integer
    End Function
    Private name As String
End Class
```

والآن من ضمن جدول خصائص ال CTS التي ستحتاج إليها فيما لو رغبت في التعامل مع CTS في الكومبايلر الخاص بك مثلاً . ستجد الخاصية Is the class abstract or concrete ، ستحتاج قبل اخبار CTS بان لديك Class هنا ان تخبره ايضاً بمثل هذه النقاط.

ملاحظة

لا تقلق لو لم تتضح لديك مثل هذه الخصائص ، في دروس قادمة سنشرح بالتفصيل كيفية عمل Class وانواعه وطرقة المختلفة.

8. 1. CTS Members

بعد ان قمت بوصف الأنواع المختلفة من خلال CTS ، يمكنك اضافة اي عدد من ال Members إلى كل منهم ، ال Members يمكن ان تكون متغيرات عادية لحقول Fields ، يمكن ان تكون ايضاً :

constructor, finalizer, static constructor, nested type, operator, method, property, indexer, field, read-only field, constant, event

كل واحد من هذه الأعضاء لديه ما يعرف باسم (visibility trait) أو مدى رؤية ، يمكن ان يكون Public مثلاً بحيث يمكن رؤيته من خارج الفئة Class .. الخ مما سنتعرف عليه لاحقاً ضمن دروسنا ، ما يعنينا هنا ان نعرف ان كل هذه الخصائص يتم توصيفها في CTS.

8. 2. CTS Data Types

يحتوي ال CTS ايضاً على ال DataTypes المدعومة من قبل net. ، هذا الجدول من كتاب Pro C# 2008 مع توضيح لل keywords للغات net. الاساسية الثلاث:

Table 1-2. *The Intrinsic CTS Data Types*

CTS Data Type	VB .NET Keyword	C# Keyword	C++/CLI Keyword
System.Byte	Byte	byte	unsigned char
System.SByte	SByte	sbyte	signed char
System.Int16	Short	short	short
System.Int32	Integer	int	int or long
System.Int64	Long	long	__int64
System.UInt16	UShort	ushort	unsigned short
System.UInt32	UInteger	uint	unsigned int or unsigned long
System.UInt64	ULong	ulong	unsigned __int64
System.Single	Single	float	Float
System.Double	Double	double	Double
System.Object	Object	object	Object^
System.Char	Char	char	wchar_t
System.String	String	string	String^
System.Decimal	Decimal	decimal	Decimal
System.Boolean	Boolean	bool	Bool

كل ما يهمك كمبرمج تطبيقات عادي ان تعرف لمعلوماتك فائدة ال CTS كنوع من المعرفة العامة لبنية net. الاساسية ، أما إذا كنت قد اعجبتك الفكرة وترغب في التعرف على المزيد عن CTS، فيمكنك البدء من هذا الرابط:

رابط

http://msdn2.microsoft.com/en-us/library/zcx1eb1e.aspx

9. Common Language Specification

كما ذكرنا في الموجز السابق ، ال CLS هي subset من ال CTS تصف الحد الأدنى من المتطلبات اللازمة لنستطيع القول ان هذه اللغة تعمل تحت بيئة net. ، وبمعنى آخر ، فإن بعض محتويات CTS اختيارية حيث يمكن ان تجدها في بعض لغات net. ولا تجدها في الآخر ، مجموعة من محتويات CTS تم تجميعها في CLS لتكون اجبارية لكل لغة تستخدم بيئة net. للتطوير.

ويتم اطلاق مصطلح CLS Rules على هذه الشروط الاجبارية ، ويتم توصيفها ولا بد من تطبيقها لكل لغة تستخدم بيئة net. للتطوير.

بالنسبة للغات #C و VB.net ، هناك مجموعة كبيرة من الخصائص التي لا يضمها ال CLS، بامكانك معرفة اذا كان الكود الذي تستخدمه موجود ضمن الCLS ام لا لمراعاة عملها على جميع أنظمة التشغيل من خلال اضافة الكود التالي أعلى الجزء الذي تود ألا يحتوي سوى على أوامر من CLS :

كود — C#

```
[assembly: System.CLSCompliant(true)]
```

كود — VB.NET

```
<Assembly: System.CLSCompliant(True)>
```

إذا كنت مهتماً بموضوع ال CLS يمكنك مواصلة القراءة من هنا:

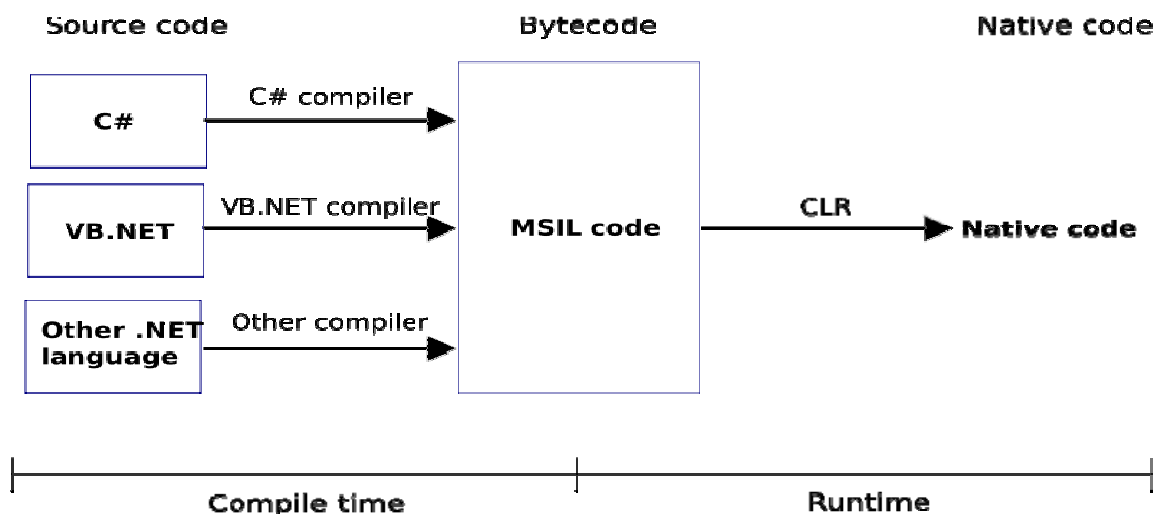
رابط

http://msdn.microsoft.com/en-us/library/12a7a7h3.aspx

10. Common Language RunTime

كما ذكرنا سابقاً ، فإن ال CLR تضم كافة المعلومات المطلوبة ليعمل كود ال .net الخاص بك على الجهاز بغض النظر عن اللغة التي تمت بها كتابة الكود الأصلي، ولتقريب المثال، إذا كنت مبرمج فيجوال بيسك فإنك مضطر لوجود msvbvm60.dll على الجهاز ليعمل برنامجك المصمم بالفيجوال بيسك وبالأدوات الرئيسية ، اما لو كنت مبرمج جافا فإنك تحتاج إلى JVM على الجهاز الذي سيعمل عليه تطبيقك ... وبنفس الطريقة ، فإن ال CLR هو الملف الوحيد الذي تحتاجه ليعمل تطبيقك المصمم تحت بيئة .net على اي جهاز كومبيوتر.

ففي عالم ال .net تحتاج لوجود مكتبة ال CLR المسماه mscoree.dll اختصاراً لـ Microsoft Common Object Runtime Execution Engine، مع بدء برنامجك ومع بداية اشارة ملف الاسمبلي إلى هذه المكتبة يقوم ال CLR بعمل Load لملف الاسمبلي وقراءة ال metadata، يعمل Load ايضاً لل Types في الذاكرة الحية، واخيراً يقوم بترجمة اوامر البرنامج الموجودة في CIL إلى platform specific instructions كما اتفقنا سابقاً ليمكنه التنفيذ على جهازك ... اخيراً يبدء ال CLR بتنفيذ برنامجك ، وخلال عمل البرنامج سيقوم الCLR بالتعامل مع الفئات الاساسية لل .net الموجودة في Base Class إذا كنت تستخدمها في برنامجك.

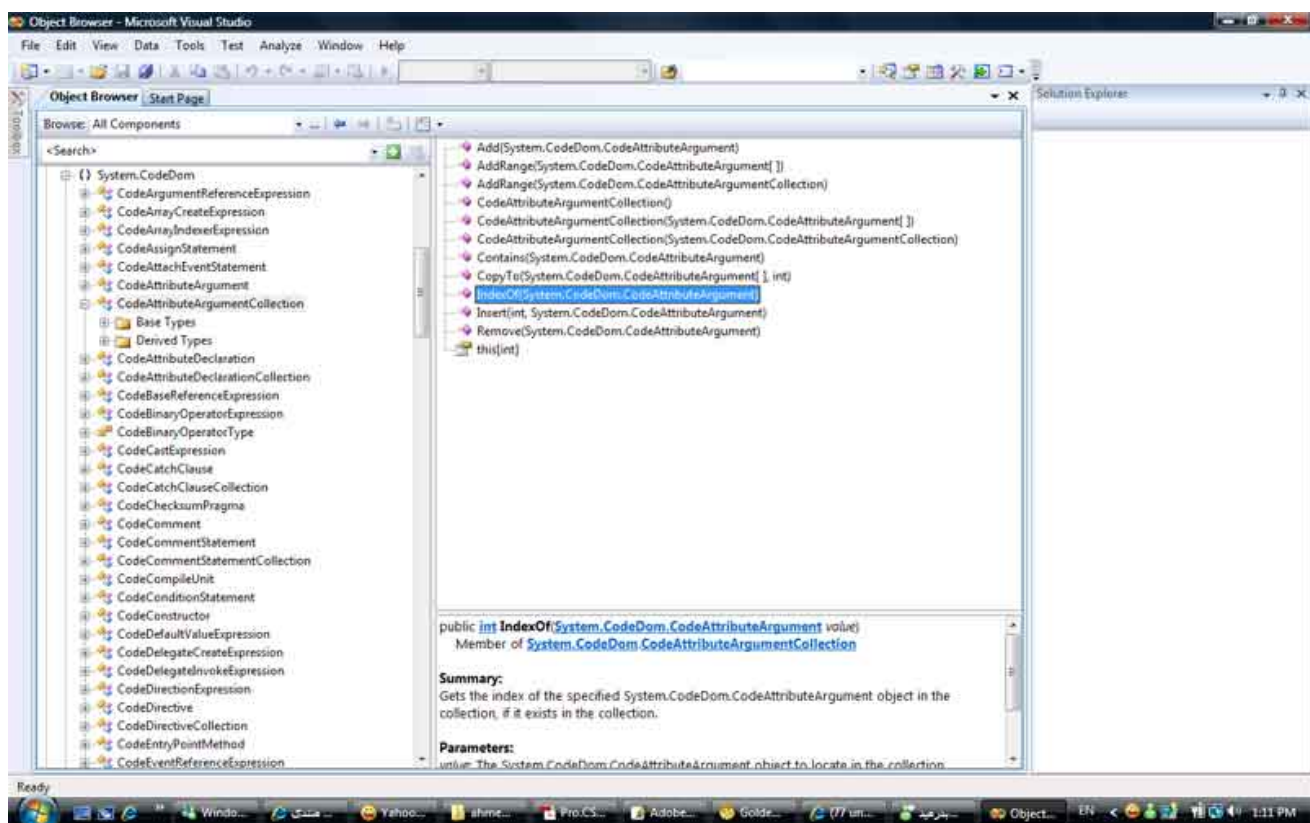
هذا المخطط من ويكبديا يوضح مراحل كتابة الكود حتى تحويله إلى native كود ومن ثم تنفيذه:

الصورة 4. 2. مراحل ترجمة الكود إلى native كود مفهوم من الآلة.

وكالعادة، إذا كنت ترغب في معرفة المزيد يمكنك البدء من هنا:

رابط

http://msdn2.microsoft.com/en-us/library/8bs2ecf4(VS.71).aspx

11. استخدام ال Object Browser

من قائمة View اختر Object Browser ، ستظهر لك الشاشة التالية:

الصورة 4. 3. ال Object Browser في الفجوال ستوديو

من خلال هذه الأداة المتوفرة مع اصدارات Visual Studio المختلفة ، تستطيع استعراض كافة المكتبات والفئات الرئيسية في .net، وتستطيع ان تعرف مكانه وملفات الاسمبلي التي يتبعها ، طريقة استخدامه ونبذة مصغرة عن فائدته.

لمزيد من المعلومات عن اي منها يمكنك اللجوء إلى مكتبات MSDN سواء الموجودة في قائمة Help، او الموجودة على الانترنت مباشرة في موقع MSDN.

12. مجالات الأسماء Name Spaces

من خلال الدروس السابقة ، تستطيع ان تلاحظ أن System هي ال namespace الرئيسية التي تجد تشتق منها أغلب ال name spaces الأخرى.

هناك name space آخر باسم Microsoft تجد تحتها العديد من الفئات مثل Microsoft.ManagementConsole وجميع الفئات المشتقة من فئة Micosoft هي خاصة فقط بالتعامل مع خدمات مايكروسوفت ولذا فهي لا تعمل تحت ال mono مثلاً.

لاحقاً، سيكون من الواضح ان تفسر ال name space التالي :
System.Windows.Forms انها تحتوي على المهام الرئيسية التي تحتاجها لبناء Forms.

12 .1. استيراد مجالات الأسماء

يمكنك استيراد مجال الاسماء اعلى ملف الكود الخاص بك بالشكل التالي

كود C#
```
using System.Data.SqlClient;
```

كود VB.NET
```
Imports System.Data.SqlClient
```

إذا قمت على سبيل المثال باستيراد مجال الاسماء كما اوضحنا ، فيمكنك كتابة الأمر التالي مباشرة:

كود | C#

```
SqlConnection sql1 = new SqlConnection();
```

كود | VB.NET

```
Dim sql1 As SqlConnection = New SqlConnection()
```

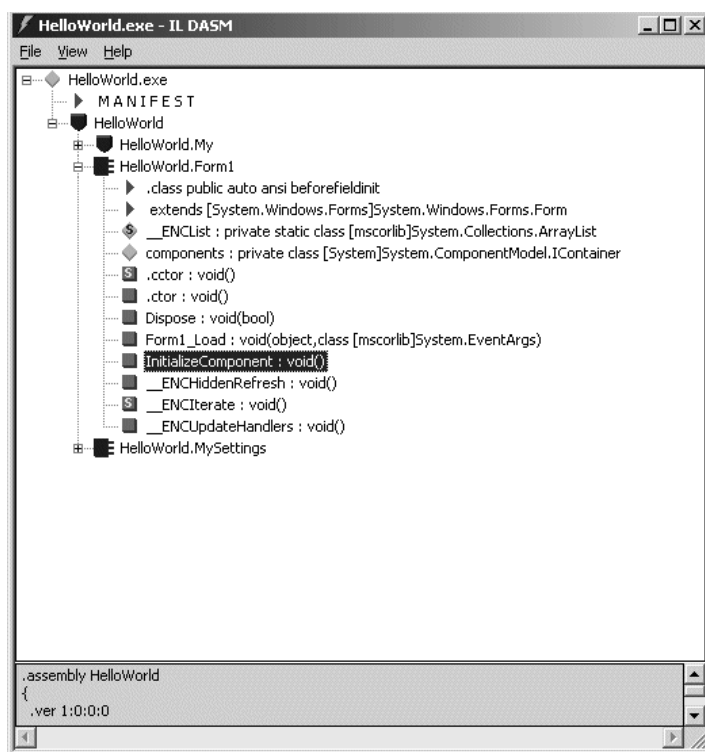
في المقابل ، لو لم تقم بتعريفه في مجال الاسماء ، ففي هذه الحالة انت مضطر لدخوله بالترتيب في كل مرة تستخدمه فيها:

كود | C#

```
System.Data.SqlClient.SqlConnection sql1 = new
System.Data.SqlClient.SqlConnection();
```

كود | VB.NET

```
Dim sql1 As System.Data.SqlClient.SqlConnection = New
System.Data.SqlClient.SqlConnectionn()
```

13. استخدام البرنامج ildasm

ضمن البرامج الملحقة مع Visual Studio 2008 تجد في الغالب برنامج ildasm ، يمكنك هذا البرنامج من الاطلاع على الاسمبلي الخاص بأي برنامج قمت بعمله باستخدام .net ، ايضاً يمكنك من رؤية ال CIL الخاص بهذا البرنامج ... هذه صورة من البرنامج :

الصورة 4. 4. برنامج ال ildasm

ويمكنك تحميل نسخة من البرنامج ومعرفة المزيد عنه من هنا :

رابط

http://msdn.microsoft.com/en-us/library/aa730858.aspx

14. هل تبحث عن Open Source .net ؟

رابط

http://www.mono-project.com

او Mono project هو اشهر ال Projects التي تتيح لل CIL العمل على توزيعات لينوكس المختلفة.

هناك مشروع آخر باسم Portable.NET يمكن CIL من العمل على منصات مختلفة. تجد رابطاً للمشروع الثاني هنا :

رابط

http://www.dotgnu.org

في أجزاء قادمة من الكتاب سنتعرف بصورة تفصيلية أكثر عن MONO.

الباب

5

الواجهة الأساسية للفيجوال ستوديو

1. كيف أكتب الكود؟

الإجابة التقليدية على السؤال السابق هي من خلال Visual Studio 2008، إلا أن هذه الاجابة ليست مكتملة تماماً ، فالإجابة الصحيحة هي انه ومن خلال وجود فقط. net framewrok 3.5 Development Kit على أي جهاز فإنك ستكون قادراً على تطوير تطبيقات .net 2008 ، سنتعرف في هذا الدرس على عجالة على بعض هذه الطرق قبل ان ننتقل للحديث حول الطريقة الاساسية التي سنتعامل معها من خلال Visual Studio 2008

مبدئياً يمكنك تحميلها من هذا الرابط

رابط

http://msdn2.microsoft.com/en-us/library/zcx1eb1e.aspx

1 .1. استخدام Visual Studio 2008 Command Prompt

لو افترضنا اننا سنقوم ببرمجة C# ، يمكنك من خلاله كتابة الأمر csc -? الإطلاع على الأوامر المطلوبة، ستكون هذه الأوامر كافية لتستطيع البدء بترجمة كود C#، لكن ستتضطر لعمل كل شيء يدوياً، ستحتاج لتحديد المخارج والاسمبلي والملفات .... الخ ، في المقابل ستستفيد من ميزة كونك تقوم بعمل كل شيء يدويا وهو ما يتيح لك فرصة ذهبية للتحكم في كل ما يتعلق بالبرنامج من الألف إلى الياء.

الصورة 5. 1. ال Visual Studio 2008 Command Prompt (تجدها في المجلد Visual Studio 2008- > Visual Studio Tools في قائمة البرامج)

بأبسط صورة ، يمكنك عمل Compile لاي ملف cs ترغب به بالشكل التالي :

كود	Command Prompt

```
csc File.cs
```

أو لتحويله إلى ملف dll:

كود	Command Prompt

```
csc /target:library File.cs
```

أو لتحويله إلى ملف exe:

كود	Command Prompt

```
csc /out:My.exe File.cs
```

اخيراً لعمل Compile لعدة ملفات موجودة في مجلد واحد :

كود	Command Prompt

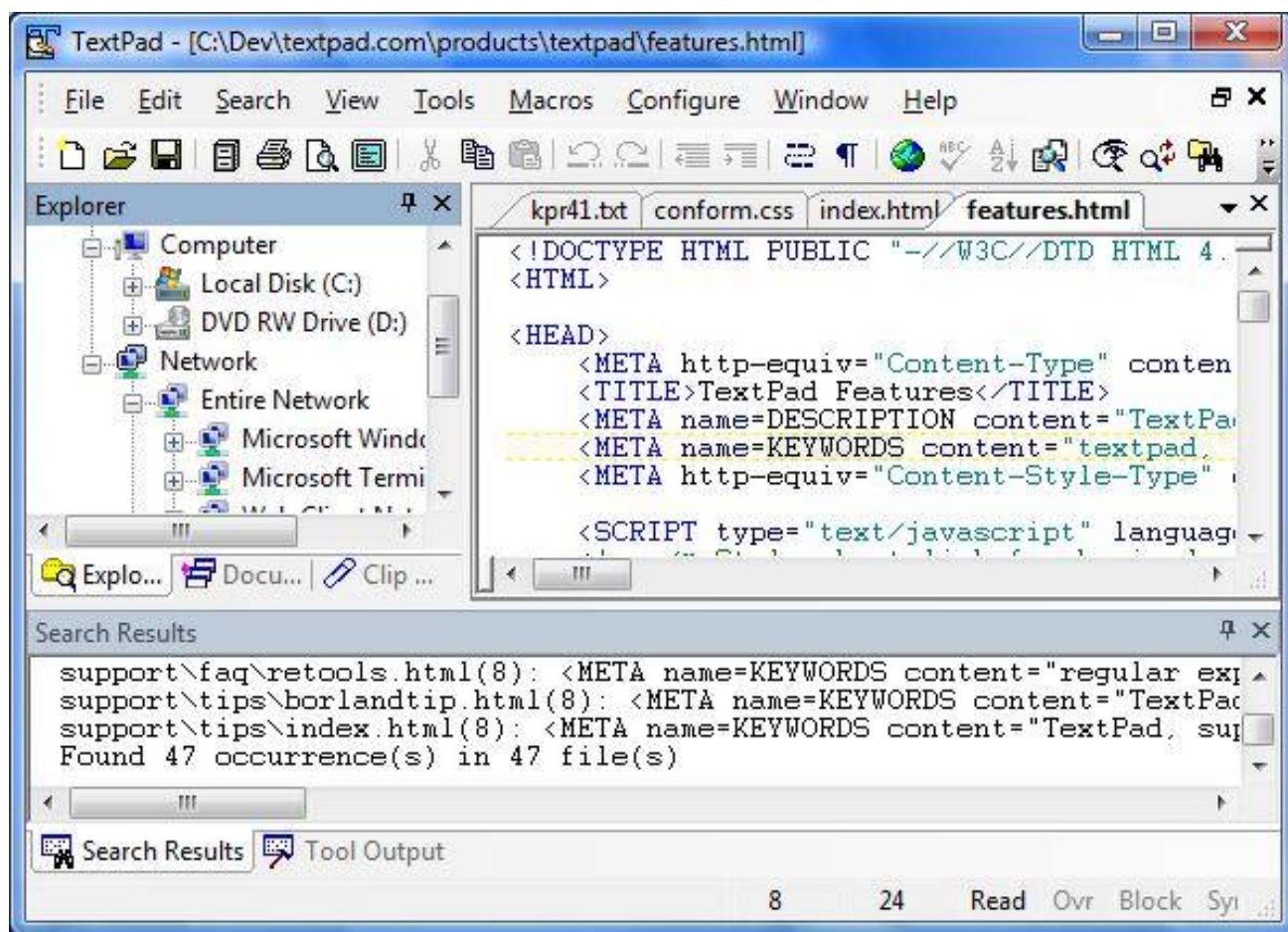
```
csc /define:DEBUG /optimize /out:File2.exe *.cs
```

يمكنك معرفة المزيد ابتداء من هذا الرابط من مايكروسوفت:

رابط

http://msdn2.microsoft.com/en-us/library/78f4aasd(VS.80).aspx

1. 2. استخدام ال TextPad

برنامج قابل لاعادة التهيئة، بمعنى انه قابل للعمل على اي شيء وقابل للتخصيص ليعمل كأي شيء، نسخة مطورة من ال Notepad قابله للتطوير والتعديل ، يمكنك استخدامها كاداة تحرير لاي لغة برمجةاو صفحات ويب اوHTML او حتى اسمبلي ... يمكنك استخدامها لتريب اوراقك ... لأي شيء ما دمت تستطيع تعديل الخصائص المطلوبة.

الصورة 5. 2. برنامج ال TextPad.

واحدة من ضمن خصائص ال TextPad انك تستطيع اضافة ملف لتعريف لغة C# او VB.net عليه، بعد اضافة الملف ستجده يقوم بتلوين الأكواد كما فيVisual Studio، يمكنك اضافة امر Compile حيث تجد نسخة من ال RE المستخدمه للغة ال C# ... الخ ، لمعرفة المزيد حول هذا الموضوع يمكنك تتبع الرابط التالي والانطلاق منه

رابط

http://www.eggheadcafe.com/community/aspnet/2/10014016/textpad-and-c.aspx

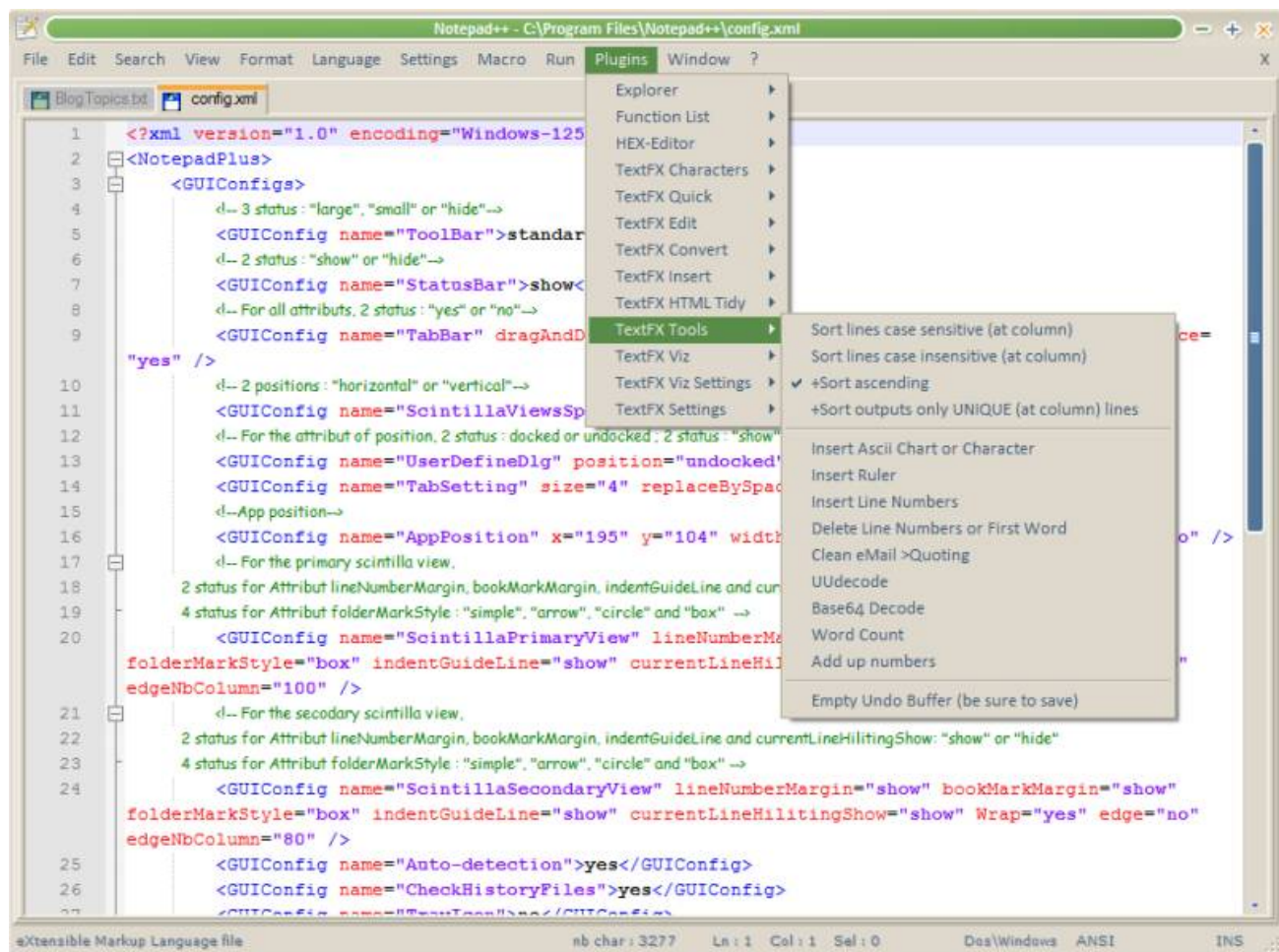
كما تستطيع ان تجد ملفات كل لغات البرمجة او الوصف لتركيبها على ال Textpad هنا

رابط

http://www.textpad.com/add-ons/cliplibs.html

بامكانك التعديل في لغة البرمجة نفسها إن اردت عن طريق تغيير الReguler Expressions، لو قمت بوضع قوانين للغة البرمجة خاصتك فباستطاعتك تنفيذها على ال Textpad ايضاً.

1. 3. استخدام ال ++Notepad

تطبيق مماثل ، إلا انه مجاني بالكامل ، يتيح لك بعض الخصائص التي ربما لا تتوفر **لل** Textpad مثل خاصية ال auto complete ، اضف لذلك انه مفتوح المصدر بمعنى انه متاح بالسورس كود.

الصورة 5. 2. برنامج ال ++Notepad.

يمكنك معرفة المزيد هنا:

 رابط

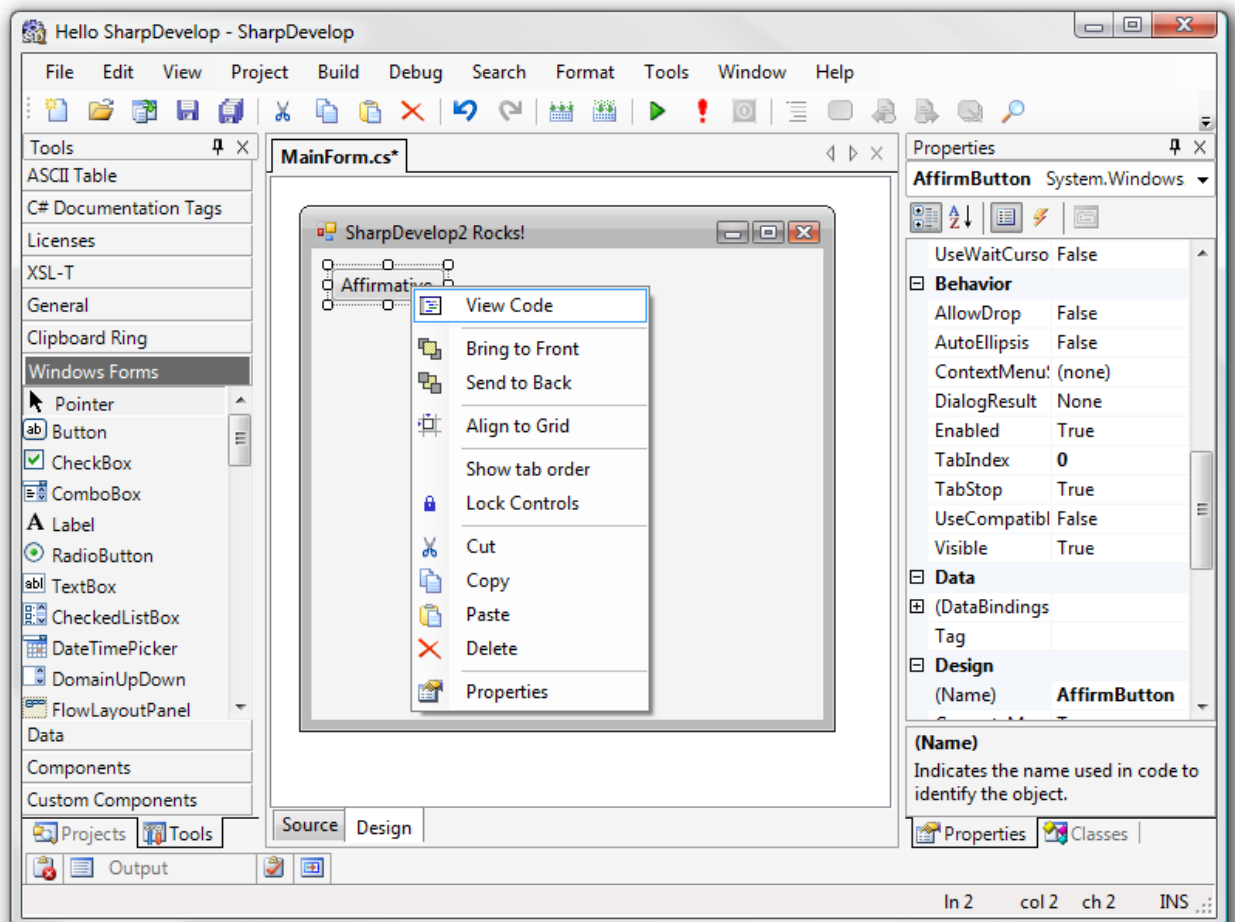
http://notepad-plus.sourceforge.net/uk/about.php

البرنامج نفسه تجده هنا مع مراحل تطويره حيث انه Open Source على هذا الرابط

 رابط

http://sourceforge.net/projects/notepad-plus/

1. 4. SharpDevelop

موجه لخدمة ال #C ، يتميز بوجود واجهة للتصميم ايضاً :

الصورة 5. 3. برنامج ال SharpDevelop.

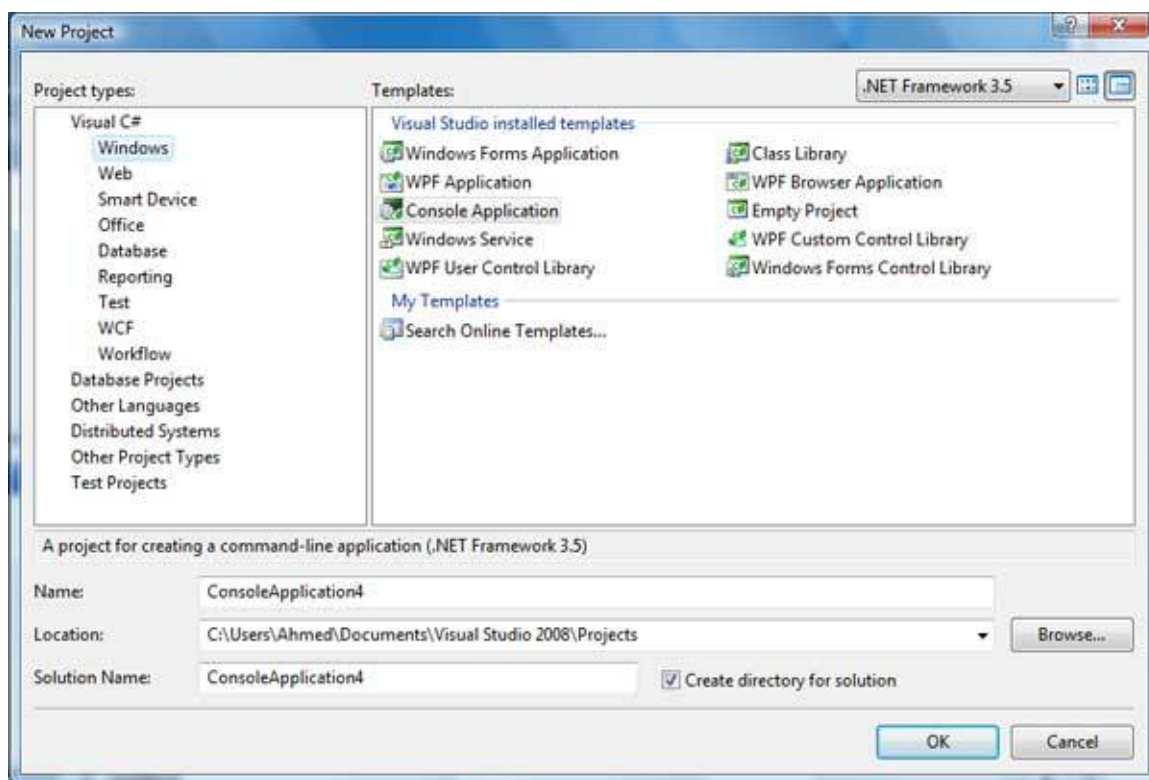
وهو الاقرب للفيجوال ستوديو في المظهر والأدوات ... يمكنك البدء بالتعرف عليه عبر الرابط التالي :

رابط

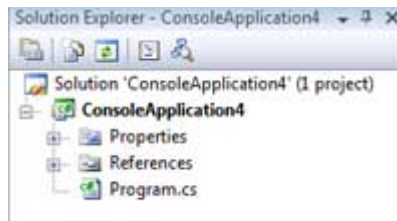
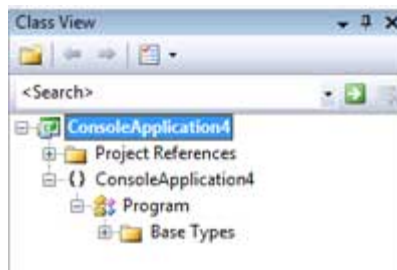
http://www.icsharpcode.net/OpenSource/SD/

2. البداية مع ال Visual Studio 2008

ال IDE المعتمد من مايكروسوفت لكتابة ال C# او ال VB.net هو ال Visual Studio باصدارتيه، حيث نجد الاصدار Proffessional وهو غير مجاني ، اما Express والذي يكون موجه للغة معينة فهو مجاني مثل Visual C# 2008 Express .

بعد تشغيل البرنامج لأول مرة ، وحسب اللغة المستخدمة ستجد أول ما تجد خيارات انشاء مشروع أو Web Site جديد اضافة للتعديل على الموجودين ، من قائمة New اختر Project مثلاً

الصورة 5. 4. انشاء مشروع جديد في ال Visual Studio 2008.

ابسط الأنواع هي Console Application وهي شاشة لكتابة الكود تحتوي مبدئياً على Class واحد باسم Main وال name space هو اسم المشروع الذي اخترته ... فقط. باقي الانواع سنتطرق لها في مراحل مختلفة من الكتاب ، سنستخدم Console لإنه ابسط الأنواع ونستطيع من خلاله فهم كامل بيئة net 2008.

على يمين الشاشة تجد ال Solution Explorer ، حيث يمكنك التنقل بين مكونات المشروع المختلفة ، تحت تبويب References تجد المكونات المضافة إلى برنامجك ، تستطيع اضافة مكونات جديدة من خلال الضغط على References بزر الفأرة الأيمن واضغط Add.

اسفل هذا التبويب يمكنك التبديل بين اسلوب العرض ، تستطيع اختيار عرض Class بحيث يتم عرض جميع ال Classes الموجودة في مشروعك والطرق والخصائص فيها :

تحت تبويب Properties تجد خصائص المشروع ، لو قمت بالضغط عليه مرتين ستجد خصائص المشروع حيث يمكنك الخصائص الاساسية لمشروعك.
يتيح لك ال net. ايضاً Class Designer حيث يمكن رسم الفئات Classes وتحديد العلاقات بينهما بصورة مرئية ومن ثم يتم تطبيقها مباشرة مع الروابط بينهما على المشروع مباشرة.

2 .1. برنامجك الأول

سنجرب اول برنامج بسيط لنا ، في ال function المسماه Main ضع الكود التالي:

كود C#

```
Console.Write("Hello World");
Console.ReadKey();
```

كود VB.NET

```
Console.Write("Hello World")
Console.ReadKey()
```

فقط ... تكون هكذا قد أنشأت برنامجك الأول في Visual Studio 2008 . جرب الضغط على زر F5 وجرب ناتج البرنامج.

الباب

6

مكونات اللغة الرئيسية

الجزئ 01

قبل الابحار في عالم المكونات الاساسية للغات الدوت نت ، سنتعرف على سوية على بعض خصائص ال Console والتي سنستخدمها لاحقاً في دروسنا.

1. خصائص Console

حتى هذه المرحلة من الدروس ، نستطيع الآن الكتابة على الشاشة وعمل دوال واستدعاءها واسترجاع قيم وطباعتها ، عند هذه المرحلة سنأخذ راحة قصيرة للتعرف على العناصر الاساسية للبيئة Console التي نعمل عليها حالياً.

اهم الدوال التي تحتويها الفئة Console هي دوال الادخال والاخراج ، وهي:

كود C#

```
// نص لكتابة .
Console.WriteLine(string)
// الإنتهاء بعد جديد لسطر الانتقال ضغط مع نص لكتابة .
string x = Console.ReadLine();
// انتر ضغط مع القراءة من الانتهاء ويتم المستخدم من مدخلات لقراءة
int x = Console.ReadKey();
// إنتر ضغط مع القراءة عملية انهاء ويتم فقط واحد حرف قراءة
ConsoleKeyInfo r = Console.ReadKey();
// زر مثل شيء اي ادخال يمكن ، إدخاله بعد القراءة من والانتهاء فقط واحد حرف قراءة
خاص متغير في الناتج تخزين يتم لذا مثلاً الأسهم
```

كود VB.NET

```
' نص لكتابة .
Console.WriteLine(String)
' الإنتهاء بعد جديد لسطر الانتقال ضغط مع نص لكتابة .
Dim x As String = Console.ReadLine()
' انتر ضغط مع القراءة من الانتهاء ويتم المستخدم من مدخلات لقراءة
Dim x As Integer = Console.ReadKey()
' إنتر ضغط مع القراءة عملية انهاء ويتم فقط واحد حرف قراءة
Dim r As ConsoleKeyInfo = Console.ReadKey()
' زر مثل شيء اي ادخال يمكن ، إدخاله بعد القراءة من والانتهاء فقط واحد حرف قراءة
خاص متغير في الناتج تخزين يتم لذا مثلاً الأسهم
```

دوال أخرى خاصة بعمليات الألوان مثل ForeGroundColor للون النص و BackgroundColor للون الخلفية، تستطيع تعيين اللون عن طريق الفئة ConsoleColor بالشكل التالي

كود C#
```
Console.ForegroundColor = ConsoleColor.Yellow;
```

ملاحظة

لو كنت مبرمج VB.net فلن تحتاج سوى لازالة الـ ; من آخر الجملة .

هناك خصائص أخرى مثل WindowWidth و Title وخلافها لتحديد مظهر النافذة.

نعود لأحد مواضيع الطباعة ، لنفترض اننا نريد القيام بطباعة النص التالي:

الإسم: المتغير name

العمر: المتغير Age

يمكننا القيام بذلك عن طريق كتابة الكود التالي:

كود C#
```
Console.Write("First Name: " + name + " - Age: " + age);
```

هناك طريقة أخرى افضل ايضاً ، بالطريقة التالية:

كود C#
```
Console.Write("First Name: {0} - Age: {1}", name, age);
```

بواسطة الطريقة الثانية ، يمكنك عمل Format للنص باستخدام رموز d للارقام و e لل exponential بالشكل التالي مثلاً:

كود C#
```
Console.WriteLine("E format: {0:E}", 99999);
```

2. تعريف المتغيرات

كما ذكرنا في الدروس السابقة يمكننا تعريف المتغيرات بالطريقة التالية

كود	C#

```
int x;
string name;
```

كود	VB.NET

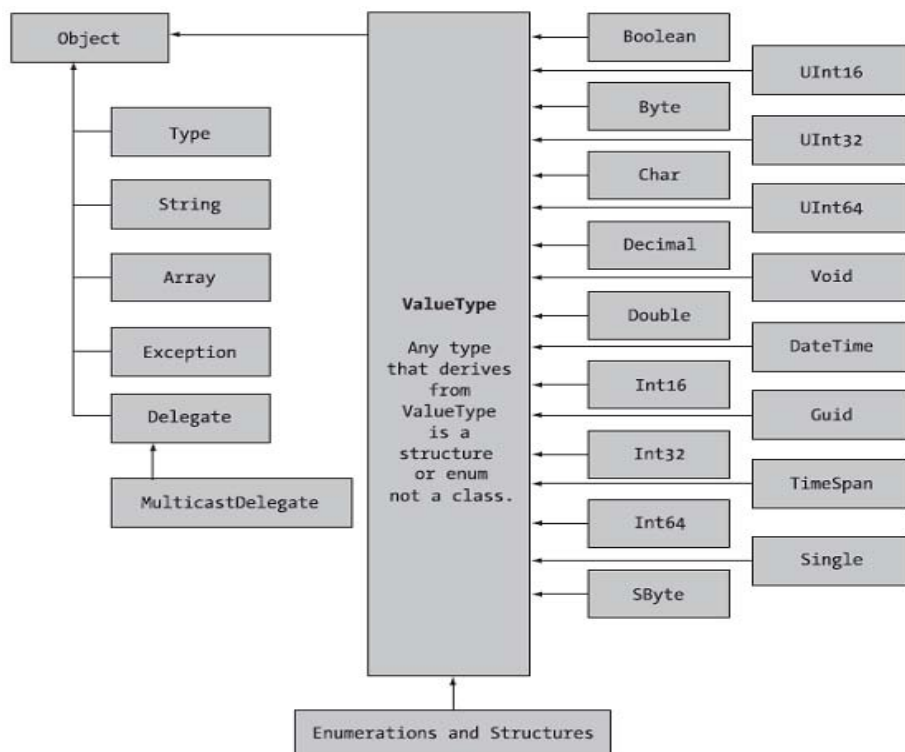
```
Dim x As Integer
Dim name As String
```

وخلافه ، ولكنك بالتأكيد تلاحظ وجود الكلمة new في كثير من تعريف المتغيرات ، فما هي مهمتها ؟

بعض الانواع البسيطة يمكن تعريفها باستخدام new وهو ما سيعيدها إلى صورتها الافتراضية الموجودة في المنشئ الخاص بها ، حيث يتم تحويل الانواع المنطقية bool إلى false والارقام إلى صفر وكذلك باقي أنواع المتغيرات.

لكن هناك انواع اخرى من البيانات لا يمكنك استخدامها إلا باستخدامnew مثل ال arraylists وObjects وخلافه مما سنتعرف عليه في حينه.

هذا هو تقسيم الانواع في C#

الصورة 6. 1. الأنواع في ال C# (الصورة من كتاب ProCSharp 2008 and .net 2008 platform)

جميع الانواع المشتقة من object تملك بالضرورة طرقه الاساسية مثل Equals و GetHashCode وخلافه.

أما الطرق الاساسية للانواع الرقمية مثل int و long فتشمل الخصائص الاساسية مثل MaxValue و ... MinValue فيما يحتوي ال char على خصائص مثل IsDigit و IsLetter لتحديد طبيعة المدخلات.

2 .1. أنواع المتغيرات

في اي جزء من البرنامج داخل ال Class يمكننا تعريف المتغيرات حسب النوع , int string مثلاً، ويتم تحديد مدى الوصول عبر المكان الموجود فيه التعريف.

لتوضيح هذه النقطة سنفترض متغير تم تعريفة داخل دالة ، هذا المتغير لن يستطيع أحد الوصول إليه إلا خلال الدالة ، ونفس الأمر لو تم تعريفه داخل شرط أو داخل حلقة تكرارية ، أما لو قمنا بتعريف متغير خارج الدالة (داخل الفئة Class مباشرة) فيمكن لاي دالة الوصول إليه مباشرة ، في هذه الحالة يسمى Member .
لدينا نوعين من المتغيرات ، النوع الأول وهو البسيط مثل int و long وخلافه حيث يتم يشير اسم المتغير لمكانه من الذاكرة أما المركب مثل Array وStruct فيشير المتغير إلى مكان تستطيع منه الاشارة إلى المتغير في الذاكرة ، سنتعرف على هذا الموضوع بالتفصيل في دروس قادمة .

2 .2. الثوابت Constants

ال Constant هو نوع من البيانات لا يمكن تغيير قيمته ابداً ، لذا يعرف باسم Constant او الثابت، يمكن كتابته بالشكل التالي:

كود C#

```
public const int myNumber = 100;
```

كود	VB.NET

```
Public Const myNumber As Integer = 100
```

يتم استخدام هذه الثوابت كقيم ثابتة لبعض المعادلات مثل PI=3.14 ، مثل رقم معين تستخدمه الشركة في الحسابات ... الخ . في هذه الحالة يفضل تعريفه كثابت بدلاً من تعريفه كمتغير لضمان استحالة تغييره في اي وقت.

يتم تعريف الثابت مرة واحدة واعطاءه القيمة لحظة تعريفه فقط...

2. 3. القيم للقراءة فقط Read Only Field

يتم تعريفه بالشكل التالي:

كود	C#

```
public readonly double PI = 3.14;
```

كود	VB.NET

```
Public ReadOnly PI As Double = 3.14
```

يعمل مثل ال Constant تماماً ، ما عدا انه يختلف في امكانية اعطاءه قيمة بعد انشاءه مرة واحدة دون ان شرط اعطاء القيمة في نفس لحظة التعريف بالشكل التالي مثلاً :

كود	C#

```
class MyMathClass
{
    public readonly double PI;
    public MyMathClass()
    {
        PI = 3.14;
    }
}
```

كود	VB.NET

```
Class MyMathClass
    Public ReadOnly PI As Double
    Public Sub New()
        PI = 3.14
    End Sub
End Class
```

3. المتغيرات النصية String

ستتحدث في درسنا هذا حول كل ما يتعلق بالنوع String

3. 1. الخصائص و الدوال الأساسية لل String

الدالة أو الخاصية	الإستخدام
Length	تحدد طول النص
Compare	للمقارنة بين نصين
Contains	للبحث عن نص او حرف ضمن النص
EndWith	لمعرفة فيما اذا كان النص يبدأ او ينتهي بحرف او نص معين
IndexOf LastIndexOf	لمعرفة مكان وجود حرف او بداية نص معين ضمن النص ، سواء من البداية او من النهاية
Remove	حذف جزء معين من النص
Insert	ادراج نص داخل ال String
Replace	استبدال جزء من النص
Split	تقسيم النص حسب شيء معين إلى مصفوفة ، مثلاً تقسيم النص مع كل علامة (-) إلى مصفوفة جديدة
ToUpper	لتحويل حالة الاحرف بين capital وsmall

الجدول 6. 1. خصائص و دوال الفئة String

3 .2. تقسيم النصوص

يمكنك استخدام الدالة Split لتقسيم محتويات النص إلى عناصر في مصفوفة، المثال التالي يوضح تقسيم نص بناء على علامات – في وسطه :

كود C#
```
string[] newarray = g.Split("-");
```

كود VB.NET
```
Dim newarray As String() = g.Split("-")
```

3 .3. دمج النصوص

الطريقة الابسط لدمج النصوص هي باستخدام + ، او باستخدام الدالة Concate بالشكل التالي:

كود C#
```
string s3 = String.Concat(s1, s2);
```

كود VB.NET
```
Dim s3 As String = [String].Concat(s1, s2)
```

3 .4. مقارنة النصوص

يمكن استخدام المعامل == لمقارنة النصوص في C# أو = في VB.net ، إلا انه من المفضل استخدام الدالة Equals بالشكل التالي:

كود C#
```
Console.WriteLine(s1.Equals(s2));
```

كود VB.NET
```
Console.WriteLine(s1.Equals(s2))
```

3 .5. Escape Characters – سي شارب فقط

في كل لغات عائلة السي ، تجد حرف / محجوزاً لبعض الحروف الخاصة مثل n/ لسطر جديد و t/ لعمل Tab وغيره ، بالشكل التالي مثلاً:

كود | C#
```
Console.Write("My Name:/nAhmed Gamal");
```

لذا إذا اردت ان تكتب / فلا بد لك ان تكتب //، هناك حل آخر وهو استخدام ما يعرف باسم Verbatim Strings حيث يمكنك في هذه الحالة كتابة نص عادي بدون القلق من ال Escape Characters بالشكل التالي مثلاً:

كود | C#
```
Console.WriteLine(@"C:\MyApp\bin\Debug");
```

3 .6. التحويل من وإلى String

تحتوي معظم الانواع الاساسية على الدالة Parse والتي تحول النص إلى مناظره ، بالشكل التالي مثلاً:

كود | C#
```
int x = int.Parse("1");
bool v = bool.Parse("True");
```

كود | VB.NET
```
Dim x As Integer = Integer.Parse("1")
Dim v As Boolean = Boolean.Parse("True")
```

كما تحتوي ايضاً على الدالة ToString لتحويلها إلى نص بالشكل التالي مثلاً

كود | C#
```
string x = m.ToString();
```

كود | VB.NET
```
Dim x As String = m.ToString()
```

3 .7 StringBuilder

عندما نقوم بدمج النصوص ، يلجأ الكثيرون منا لاستخدام المعامل + في C# أو المعامل & في VB.net بالشكل التالي - مثال اضافة امتداد البريد الإلكتروني:

كود | C#

```
// C#:
Label1.Text = Text1.Text + "@hotmail.com";
```

كود | VB.NET

```
'VB.net:
Label1.Text = Text1.Text + "@hotmail.com"
```

إلا أن Visual Studio قدمت لنا طريقة أخرى لدمج النصوص باستخدام الفئة StringBuilder، في مجال الأسماء System.Text.StringBuilder، يمكن استخدامها لدمج النصوص بالشكل التالي:

كود | C#

```
//C#:
System.Text.StringBuilder mail = New System.Text.StringBuilder(Text1.Text);
mail.Append("@hotmail.com");
```

كود | VB.NET

```
'VB.net:
Dim mail As New System.Text.StringBuilder(Text1.Text)
mail.Append("@hotmail.com")
```

الفارق بين الاثنين يكمن في ان StringBuilder يظل كما هو منذ لحظة انشاءه، اما استخدام المعاملات + أو & فهو يقوم بعمل Object جديد من ال String مع كل عملية دمج.

والآن سنفترض مثال Loop تقوم بدمج عدد من النصوص، وسنصور المقارنة مباشرة من كتاب net Gotachas.، حيث قام بعمل اختبار لعدد عمليات دمج ابتداء من 10 عمليات دمج وحتى 1000000 عملية دمج ، وقام بمقارنة الأداء بين استخدام StringBuilder او استخدام معاملات الدمج التقليدية ، مع ذكر زمن التنفيذ بالثانية لكل منهم.

Table 1-3. Performance, in seconds, of concatenation versus StringBuilder

# of appends	+	StringBuilder
10	0.000	0.00
100	0.000	0.00
1,000	0.000	0.00
2,500	0.000	0.00
5,000	0.020	0.00
7,500	0.050	0.00
10,000	0.090	0.00
15,000	0.250	0.00
25,000	1.052	0.00
35,000	2.373	0.00
50,000	5.699	0.00
65,000	10.625	0.00
75,000	14.831	0.01
85,000	19.418	0.01
100,000	27.159	0.01
150,000	65.374	0.01
250,000	209.221	0.02
350,000	441.615	0.02
500,000	910.129	0.04
650,000	1521.708	0.06
750,000	1999.305	0.06
850,000	2576.575	0.06
1,000,000	3562.933	0.07

الصورة 6. 2. مقارنة نتائج سرعة تنفيذ النتائج بين المعامل + و ال StringBuilder

لا تنسى ان 3562.933 ثانية تعني 59.4 دقيقة تقريباً...

والآن ... هل ما زلت تستخدم & او + لدمج النصوص ؟؟؟

توفر الفئة StringBuilder ايضاً عمليات استبدال Replace وادراج Insert وحذف Remove وعمليات نصوص أخرى كثيرة ، يمكنك الاطلاع على أوجه كثيرة للمقارنة مدعومة بالرسوم البيانية من خلال هذا الرابط من Code Project:

رابط

http://www.codeproject.com/KB/cs/StringBuilder_vs_String.aspx

4. التعامل مع التاريخ و الوقت

تجد جميع ما يتعلق بالتاريخ والوقت في المكتبة DateTime ، فمثلاً لاضافة تاريخ معين:

كود C#
```
DateTime dt = new DateTime(2004, 10, 17);
```

لطباعة التاريخ الحالي:

كود C#
```
Console.WriteLine(DateTime.Now);
```

يمكن التعامل مع الوقت أيضاً باستخدام TimeSpan

كود C#
```
TimeSpan ts = new TimeSpan(4, 30, 0);
```

الاضافة والطرح باستخدام الداول Add و Substract بالشكل التالي:

كود C#
```
Console.WriteLine(ts.Subtract(new TimeSpan(0, 15, 0)));
```

أو

كود C#
```
dt = dt.AddMonths(2);
```

ملاحظة

لو كنت مبرمج VB.net فلن تحتاج سوى لازالة الـ ; من آخر الجملة .

5. التحويل بين المتغيرات المختلفة

بداية ، تنقسم التحويلات بين المتغيرات إلى نوعين رئيسين:

- Widening Conversions
- Narrowing Conversions

5. 1. Widening Conversions

يقصد بهذا النوع من التحويلات تلك التحويلات التي لا يمكن فيها خسارة اي نوع من البيانات، وتسمى باسم upward cast ، مثال ذلك التحويل من Short إلى Integer في المثال التالي :

كود	C#

```
short x = 5;
power(x);
```

كود	VB.NET

```
Dim x As Short = 5
power(x)
```

وفي الدالة power

كود	C#

```
int power(int number)
{
    return number ^ 2;
}
```

كود	VB.NET

```
Private Function power(ByVal number As Integer) As Integer
    Return number Xor 2
End Function
```

لو لاحظت ستجد ان الدالة تستقبل بيانات من نوع int فيما ارسلنا لها بيانات من نوع short، في هذه الحالة لن تكون هناك مشكلة لإن مدى ال Integer اكبر من مدى ال Short، وبالتالي فإن اي مدى لل Short يقع ضمن ال Integer بكل تأكيد.

5. 2. Narrowing Conversions

الحالة العكسية ، التحويل من الاكبر إلى الأصغر ، مثلاً لو كان المتغيران من نوع Integer والناتج من نوع Short، سيعمل الامر بصورة صحيحة لو كان مجموع الرقمين Integer

اصغر من الحد الأقصى لل Short، ولكن لو افترضنا ان مجموعهم تجاوز حدود مدى ال Short فإن ذلك سينتج مشكلة.

لهذا السبب ، يمنعك المترجم مباشرة من كتابة مثل هذا الكود ويعطيك رسالة الخطأ التالية:

Cannot implicitly convert type 'int' to 'short'.

لكن لو رغبنا في التحويل رغماً عن هذه النقطة حتى لو نتج عن ذلك ضياع بعض البيانات ، في هذه الحالة نلجأ لما يسمى بـ Cast

5 .3. عمليات ال Cast

لا تنطبق عمليات ال cast على التحويل من اكبر لأصغر فقط ، بل يمكن استخدامها في كل عمليات التحويل ، ابسط استخدام لها هو لتحويل Integer إلى Short بالشكل التالي:

كود C#
```
int var = 10;
short var2 = (short)var;
```

كود VB.NET
```
Dim var As Integer = 10
Dim var2 As Short = CShort(var)
```

في المثال السابق ، لو قمنا بطباعة نتيجة var2 سنجد انها 10 ، لكن ماذا لو افترضنا المثال التالي:

كود C#
```
int var = 100000;
short var2 = (short)var;
Console.WriteLine(var2);
Console.ReadKey();
```

كود VB.NET
```
Dim var As Integer = 100000
Dim var2 As Short = CShort(var)
Console.WriteLine(var2)
Console.ReadKey()
```

النتيجة لن يمكن توقعها بالنسبة للعميل، ولكن بصورة مقربة لك كمبرمج ستكون النتيجة هي - 31072، حيث سيقوم المترجم بعكس الإشارات ابتداء من 32768 ومن ثم انقاص رقم مع كل زيادة عن الرقم السابق، او باختصار فهو يقوم بطرح الرقم الناتج من الحد الاقصى او المدى الاقصى للنوع وهو 32767.

باستخدام هذا النوع من التحويلات ، فإنه من المفضل دائماً استخدام Try Catsh والتي سنتعرف عليها بالتفصيل في جزء قادم من الكتاب.

5. 4. التحويل باستخدام Convert

يمكن التحويل بين اي انواع من البيانات باستخدام الفئة Convert بالشكل التالي مثلاً:

كود C#

```
myByte = Convert.ToByte(myInt);
```

كود VB.NET

```
myByte = Convert.ToByte(myInt)
```

6. الجمل الشرطية في net.

6. 1. أساسيات الشروط

ابسط الجمل الشرطية هي تلك التي تستخدم if else، وطريقة كتابتها بالشكل التالي:

كود C#

```
if (x == 5)
    Console.WriteLine("five");
else
    Console.WriteLine("notFive");
```

كود	VB.NET

```
If x = 5 Then
    Console.WriteLine("five")
Else
    Console.WriteLine("notFive")
End If
```

يتم استخدام كافة انواع المقارنات == و =! و < و > في الجمل الشرطية ، يمكن دمج اكثر من شرط باستخدام else if بالشكل التالي:

كود	C#

```
if (x > 90)
{
     Console.WriteLine("ممتاز");
}
else if (x >= 50)
{
     Console.WriteLine("ناجح");
}
else
{
     Console.WriteLine("راسب");
}
```

كود	VB.NET

```
If x > 90 Then
    Console.WriteLine("ممتاز")
ElseIf x >= 50 Then
    Console.WriteLine("ناجح")
Else
    Console.WriteLine("راسب")
End If
```

يتم تطبيق الجملة الأولى في حالة كون الناتج (True) وإلا يتم تنفيذ الشرط الثاني. النظام القديم للغات السي والتي كانت تقضي بانها صحيحة لو كانت تساوي 1 ايضاً مثل الجملة التالية:

تنبيه

كنا قد تعودنا مع عائلات اللغات التي تتبع ال C أن نقوم بكتابة بعض شروطنا بالشكل التالي :

كود	C#

```
if (string.lenght)
```

حيث يمكن لو كان له قيمة أن يحقق الشرط = True هذا النظام لم يعد صالحاً مع السي شارب بعد ...

6 .2 دمج الشروط

لعمل اكثر من شرط يمكن استخدام && أو And للدمج بين الشروط بحيث يتم تنفيذ الشرط في حالة كونهم جميعاً True او استخدام معامل Or والذي يتم كتابته في سي شارب بالشكل التالي || فيعطي نتيجة في حالة كون اي واحد منهم صحيحاً ، المعامل Not والذي يكتب بالشكل التالي ! فيعني في حالة عدم (نفي) لاتنس ترتيب الاقواس في هذه الحالات حتى لا تتداخل الشروط بالشكل التالي مثلاً:

كود C#

```
if ((x < 90 || x > 50) && (!name = "ahmed")) ;
```

كود VB.NET

```
If (x < 90 Or x > 50) And (Not Name = "ahmed") Then

End If
```

6 .3 AndAlso

كنا مع عادتنا في برمجيات .net وفي المعامل And ان يتم اختبار الشرطين، ويكون لها جدول الناتج التالي :

مدخل 1	مدخل 2	الناتج
True	True	True
False	True	False
True	False	False
False	False	False

الجدول 6 .2. جدول الحقيقة الخاص بالمعامل And

وبرغم انك تلحظ بإنه في حالة كون الطرف الأول false فإن الجملة مباشرة ستكون خاطئة، ولا يوجد أي داعي للتحقق من الجزء الثاني من الشرط ، ولكن للاسف هذا ما لا يفعله المعامل And.

لكن مع AndAlso فالموضوع مختلف ، فهو يعيد false مباشرة في حالة وجود اي تعبير خاطئ .

ماذا نستفيد من ذلك ؟ ببساطة شديدة ، لنفترض الكود التالي :

كود C#

```
if (id > 0 & SearchForID(id) > 0)
{
    // do something
}
```

كود VB.NET

```
If id > 0 And SearchForID(id) > 0 Then
    ' do something
End If
```

في الواقع وحتى لو كان ال Id اصغر من الصفر ، فسيتم تنفيذ امر البحث وهو ما ينتج بطء شديد جداً في المعالجة لا نحتاج إليه ، يتم التغلب عليها بالطريقة التقليدية بالشكل التالي :

كود C#

```
if (id > 0)
{
    if (SearchForID(id) > 0)
    {
        // do something
    }
}
```

كود VB.NET

```
If id > 0 Then
    If SearchForID(id) > 0 Then
        ' do something
    End If
End If
```

وبطريقة AndAlso :

كود C#

```
if (id > 0 && SearchForID(id) > 0)
{
    // do something
}
```

كود VB.NET

```
If id > 0 AndAlso SearchForID(id) > 0 Then
    ' do something
End If
```

6. 4. OrElse

كما هو الحال مع And ، يتكرر الأمر مع المعامل Or والذي له جدول النتائج التالي :

مدخل 1	مدخل 2	الناتج
True	True	True
False	True	True
True	False	True
False	False	False

الجدول 6. 2. جدول الحقيقة الخاص بالمعامل Or

وبرغم انك تلحظ بإنه في حالة كون الطرف الأول True فإن الجملة مباشرة ستكون صحيحة، ولا يوجد أي داعي للتحقق من الجزء الثاني من الشرط ، ولكن للاسف هذا ما لا يفعله المعامل Or.

من اجل هذا كان المعامل OrElse والذي لا يكلفك عناء التحقق من عدة مدخلات كما هو الحال في الدرس السابق ايضاً مع AndAlso .

6. 5. استخدام ال switch

طريقة اخرى للجمل الشرطية هي استخدام switch بالشكل التالي:

كود	C#

```
switch (x)
{
    case 90:
        Console.WriteLine("ممتاز");
        break;
    case 50:
        Console.WriteLine("ناجح");
        break;
}
```

كود | VB.NET

```
Select Case x
    Case 90
        Console.WriteLine("ممتاز")
    Case 50
        Console.WriteLine("ناجح")
End Select
```

لا تنس اضافة break في اخر الشرط ، السبب ان ذلك يمنع المترجم من الاستمرار في اختبار باقي الشروط عندما يعثر على اول شرط ، اما إذا كنت ترغب في مروره على كل الشروط حتى مع تحقق اي منهم فلا تضف break ، في الفيجوال بيسك لن تتعرض لهذه المعضلة .

7. الحلقات التكرارية

نستخدم الحلقات التكرارية لتكرار سطر او امر عدة مرات ، سنشرح طرق التكرار في هذا الدرس على عجالة:

7. 1. For-Next Loop

حلقة تكرار يتم تحديد نقطة البداية والنهاية ومقدار الخطوة فيها ، ابسط مثال عليها الكود التالي

كود | C#

```
for (int i = 0; i < 10; i++)
{
    Console.WriteLine(i);
}
```

كود | VB.NET

```
For i As Integer = 0 To 9
    Console.WriteLine(i)
Next
```

يمكن عمل مقدار القفز = 2 مثلاً بالشكل التالي - لطباعة الارقام الزوجية مثلاً: -

كود | C#

```
for (int i = 0; i < 10; i += 2)
{
    Console.WriteLine(i);
}
```

كود	VB.NET

```
For i As Integer = 0 To 9 Step 2
    Console.WriteLine(i)
Next
```

2 .7. While Loop

حلقة تكرار تستمر في العمل حتى تحقق شرط معين ، لنفترض مثلاً حتى وصول العداد إلى رقم 10:

كود	C#

```
int x = 0;
while (x < 10)
{
    Console.WriteLine(x);
    x++;
}
```

كود	VB.NET

```
Dim x As Integer = 0
While x < 10

    Console.WriteLine(x)
    X+=1
End While
```

حالة اخرى غير رقمية ، ادخال بيانات حتى ادخال "exit" ، في هذه الحالة سوف نضيف Do في البداية ونضع While في النهاية.

كود	C#

```
string inp;
do
{
    inp = Console.ReadLine();
    Console.WriteLine(inp);
}
while (inp != "exit");
```

كود	VB.NET

```
Dim inp As String
Do
    inp = Console.ReadLine()

    Console.WriteLine(inp)
Loop While inp <> "exit"
```

7 .3. For Each Loop

للدوران على (كل) شيء معين، مثلاً للدوران على كل عناصر مصفوفة معينة مثلاً:

كود	C#

```
int[] arr = { 10, 20, 30, 40 };
foreach (int i in arr)
    Console.WriteLine(i);
```

كود	VB.NET

```
Dim arr As Integer() = {10, 20, 30, 40}
For Each i As Integer In arr
    Console.WriteLine(i)
Next
```

الباب

7

مكونات اللغة الرئيسية

الجزئ 02

1. لنفهم برنامجنا الأول

لو لاحظنا برنامجنا الأول لطباعة جملة Hello World والذي كان بالشكل التالي:

كود	C#

```
using System;
using System.Collections.Generic;
using System.Text;
namespace ConsoleApplication4
{
 class Program
 {
      static void Main(string[] args)
      {
          Console.Write("Hello World");
          Console.ReadKey();
      }
 }
}
```

أما في فيجوال بيسك فكان بالشكل التالي :

كود	VB.NET

```
Module Module1
    Sub Main()
         Console.Write("Hello World")
         Console.ReadKey()

    End Sub
End Module
```

اول ما سنجده في C# هو استيراد المكتبات التي سنستخدمها لكتابة اكوادنا باستخدام using أما في الفيجوال بيسك فتم تعريف Module قمت بكتابة الأكواد داخله مباشرة ، ال Module هو فئة Class عادية ما عدا أن جميع فئاتها هي Shared-static ينشأها الفيجوال ستوديو لك في حالة كون مشروعك من نوع VB.

الجزء الثاني في كود C# هو تعريف ال name space الخاص ببرنامجنا وهو الاسم الذي توضع تحته كل مكونات البرنامج ، ومن ثم ال Class الاساسي لدينا باسم Main ... كما تعلمنا في اي مبادئ للبرمجة كائنية التوجه فإن أي برنامج يتكون من واحد او اكثر من ال Classes، الدالة الرئيسية في ال Class المسمى Main هي الدالة main والتي يتم تنفيذها اول شيء في البرنامج. تستقبل الدالة Main مجموعة من الباميتير تحت اسم args ، معنى هذا ان البرنامج

يستقبل مع تشغيله مصفوفة من args[0] إلى args[n]، يمكن ارسال كل ما تريد إلى البرنامج من خلالها ، وهو ما يسمى Command Line Parameters.

لكي لا نبعد عن مسارنا في الشرح ، ال Command Line Args هي مجموعة من المتغيرات التي يمكن تمريرها للبرنامج وقت تشغيله لتتحكم في بعض النقاط ، مثلاً لو قمنا بكتابة الامر التالي:

كود	Shell

```
Explorer http://www.vb4arab.com
```

اول ما يتم فتح برنامج Explorer ، يقوم بقراءة النص الممرر له ، إذا كان موقع انترنت يقوم بفتحه بالشكل المعهود لمتصفح الانترنت، اما لو كان مسار مثلاً \:C فسيقوم بفتح متصفح المجلدات مع انهم نفس البرنامج في النهاية .

تستطيع تجربة ذلك على برنامجك من خلال قراءة المتغيرات الممررة له عن طريق For Loop

كود	C#

```
for (int i = 0; i < args.Length; i++)
    Console.WriteLine(args[i]);
```

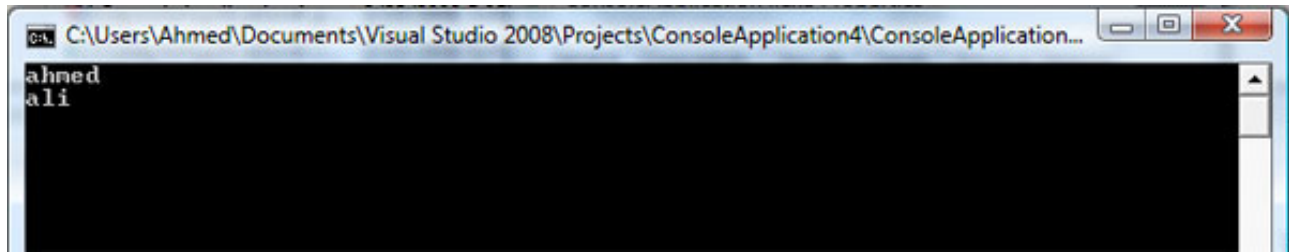
كود	VB.NET

```
For i As Integer = 0 To args.Length - 1
    Console.WriteLine(args(i))
Next
```

الآن قم بالذهاب إلى Run، قم بكتابة مسار برنامجك ومن ثم مسافة ومن ثم كتابة البارميترات التي ترغب في تمريرها، في جهازي كتبت الامر التالي في Run:

كود	Shell

```
C:\Users\Ahmed\Documents\Visual Studio 2008\Projects\ConsoleApplication4\ConsoleApplication4\bin\Debug\consoleapplication4.exe ahmed ali
```

الناتج الذي ظهر لي كان بالشكل التالي:

الصورة 7. 1. نتائج تشغيل الأمر في الشل

كتطبيق سريع ، سنقوم بقراءة المتغير الأول وفي حالة وجود 0 مثلاً يتم تلوين خلفية الكلام بالازرق ، وفيما عدا ذلك يتم تلوين خليفة الكلام بالأحمر ، هذا هو الكود الخاص بذلك:

كود C#

```
if (args[0] == "0")
{
    Console.BackgroundColor = ConsoleColor.Blue;
    Console.WriteLine("0 enterd");
}
else
{
    Console.BackgroundColor = ConsoleColor.Red;
    Console.WriteLine("Empty");
}
Console.ReadKey();
```

كود VB.NET

```
If args(0) = "0" Then
    Console.BackgroundColor = ConsoleColor.Blue

    Console.WriteLine("0 enterd")
Else
    Console.BackgroundColor = ConsoleColor.Red
    Console.WriteLine("Empty")
End If
Console.ReadKey()
```

تواجهنا هنا مشكلة، ماذا لو لم نقم بتمرير اي متغيرات .. ستظهر لك رسالة خطا ، إذن نحن بحاجة للتأكد من وجود بارمتيرس قبل قراءتها ، سنقوم بذلك عن طريق التأكد من وجود متغيرات ، إذا كان هناك فيتم التأكد من كونها صفر ، إذا اختل اي من الشرطين نلجأ للون الثاني:

كود	C#

```
if (args.Length > 0)
{
    if (args[0] == "0")
    {
        Console.BackgroundColor = ConsoleColor.Blue;
        Console.WriteLine("0 enterd");
    }
}
else
{
    Console.BackgroundColor = ConsoleColor.Red;
    Console.WriteLine("Empty");
}
Console.ReadKey();
```

كود	VB.NET

```
If args.Length > 0 Then
    If args(0) = "0" Then
        Console.BackgroundColor = ConsoleColor.Blue
        Console.WriteLine("0 enterd")
    End If
Else
    Console.BackgroundColor = ConsoleColor.Red
    Console.WriteLine("Empty")
End If
Console.ReadKey()
```

هناك حل آخر ، ماذا لو احببنا ان نجعل التأكد في سطر واحد بالشكل التالي

كود	C#

```
if (args.Length >0 && args[0] == "0")
```

كود	VB.NET

```
If args.Length > 0 AndAlso args(0) = "0" Then
```

هذا يعني انه سيتأكد من وجود بارميترس ، إذا وجدها سينتقل للشرط الثاني وإذا لم يجدها فإنه يخرج مباشرة دون قراءة الشرط الثاني...

الجدير بالذكر ان المعامل And لن يمكن تطبيقه في VB.net، بل ستضطر لاستخدام AndAlso الذي سبق شرحه في دروس سابقة.

2. الدوال Functions

كما شاهدنا في دالة Main، يمكننا انشاء اي عدد من الدوال بأي عدد من المتغيرات ، كل ما نحتاج إليه هو معرفة البارمتيرات التي نريد لها ان تصل للدالة ، وايضاً الناتج الخارج منها، لنفترض مثلاً اننا نصمم دالة لعملية الجمع ، تستقبل رقمين int وتخرج ناتج int ، ستكون الدالة بالشكل التالي:

كود	C#

```
static int sum(int number1, int number2)
{
    int total = number1 + number2;
    return total;
}
```

كود	VB.NET

```
Private Function sum(ByVal number1 As Integer, ByVal number2 As Integer) As
Integer
    Dim total As Integer = number1 + number2
    Return total
End Function
```

ومن ثم ، سنحاول من خلال Main استدعائها بالشكل التالي مثلاً:

كود	C#

```
int result = sum(5, 8);
Console.Write(result);
Console.ReadKey();
```

كود	VB.NET

```
Dim result As Integer = sum(5, 8)
Console.Write(result)
Console.ReadKey()
```

لماذا تم تعريف sum على انها static في السي شارب ؟؟

السبب انه لا يمكن استدعاء دالة غير static من خلال دالة static ، وبما ان دالة Main هي من نوع static فلا بد من ان تكون اي دالة اخرى يتم استدعائها من خلال ال main هي static ايضاً ، سنتعرف على بعض التفاصيل الاضافية لاحقاً .

3. الطرق Methods

الطرق هي الدالة التي لا تعيد ناتج ، مثل دالة عرض كتابة رسالة بالشكل التالي:

كود C#

```
void printmsg(string msg)
{
    Console.WriteLine(msg);
}
```

في الفجيوال بيسك 6، كان يتم تعريف الطريقة باسم Sub، وهو المستمر ايضاً مع VB.net ليكون بالشكل التالي:

كود VB.NET

```
Private Sub printmsg(ByVal msg As String)
    Console.WriteLine(msg)
End Sub
```

4. الوظيفة out

كما تعلمنا في المثال السابق فإننا نقوم بارجاع النتيجة من خلال return، ماذا لو اردنا اعادة نتيجة من دالة void، يتم ذلك باستخدام الدالة out بالشكل التالي:

كود C#

```
static void sum(int number1, int number2, out int total)
{
    total = number1 + number2;
}
```

كود VB.NET

```
Private Shared Sub sum(ByVal number1 As Integer, ByVal number2 As Integer,
ByRef total As Integer)
    total = number1 + number2
End Sub
```

الدالة السابقة الخاصة بالفيجوال بيسك هي الترجمة الصحيحة لكود ال C# ، ولكن لكي تعمل معك بصورة صحيحة في Module لا بد أن تستغني عن Shared ، لذا سنقوم بالاستغناء عنها ابتداء من هذا الكود حتى يحين موعد شرح معناها وفائدتها .

وعند استدعاء الدالة يتم تمرير المتغير الذي نحتاج إليه لعرض النتيجة ايضاً:

كود	C#

```
int result;
sum(5, 8,out result);
Console.Write(result);
Console.ReadKey();
```

كود	VB.NET

```
Dim result As Integer
sum(5, 8, result)
Console.Write(result)
Console.ReadKey()
```

ولكن ما هو السبب الذي قد يدفعني لاستخدام هذه الطريقة بدلاً من استخدام return؟

الاجابة الابسط، هي انني لو اردت اعادة اكثر من نتيجة مثل ناتج الضرب والجمع والقسمة ، فليس امامي حل سوى اعادة مصفوفة بالارقام وقراءتها هناك ، الحل الابسط هو باستخدام out.

5. الارسال بالمرجع byref و الارسال بالقيمة byval

لنفترض المثال الخاص بعملية الطرح بالشكل التالي:

كود	C#

```
int x = 5;
int y = 10;
sub(x, y);
```

كود	VB.NET

```
Dim x As Integer = 5
Dim y As Integer = 10
sub(x, y)
```

والدالة:

كود	C#

```
static void sub(int number1, int number2)
{
    number1 = number1 - number2;
    return number1;
}
```

كود	VB.NET

```
Private Sub [sub](ByVal number1 As Integer, ByVal number2 As Integer)
    number1 = number1 - number2
    Return number1
End Sub
```

في الدالة السابقة قمنا بتغيير قيمة number1 ، ولكن هل سيغير هذا من قيمة X التي ارسلناها، الإجابة هي بلا ، هذا ما يعرف بارسال القيمة حيث نقوم في هذه الحالة بارسال قيمة X إلى الدالة وليست قيمة X.

الارسال بالمرجع ByRef هي الحالة الثانية ، في هذه الحالة يتم ارسال عنوان المتغير X في الذاكرة إلى الدالة ، هذا ما يعني ان اي تغيير في number1 سيؤثر بالضرورة على المتغير X.

لكتابة الكود السابق بطريقة ByRef نكتب الكود التالي:

كود	C#

```
static void sub(ref int number1, ref int number2)
{
    int result = number1 - number2;
    return result;
}
```

كود	VB.NET

```
Private Sub [sub](ByRef number1 As Integer, ByRef number2 As Integer)
    Dim result As Integer = number1 - number2
    Return result
End Sub
```

الارسال بالقيمة يمكن تطبيقه فقط مع المتغيرات البسيطة، اما المتغيرات المركبة مثل Struct فيتم التعامل معه ByRef افتراضياً...

6. المصفوفات Arrays

في هذا الدرس سوف نتعرف على المصفوفات وكيفية استخدامها وتعريفها.

6. 1. ما هي المصفوفات Arrays

المصفوفة هي عبارة عن سلسلة من البيانات من نفس النوع ، لتعريف Array من الارقام طولها 5 عناصر نكتب الكود التالي:

كود C#
```
int[] intarray = new int[5];
```

كود VB.NET
```
Dim intarray As Integer() = New Integer(4)
```

يبدأ الترقيم في المصفوفات من الصفر وحتى 4، لقراءة احد عناصر المصفوفة نكتب كود مثل التالي:

كود C#
```
Console.WriteLine(intarray[3]);
```

كود VB.NET
```
Console.WriteLine(intarray(3))
```

ولقراءة جميع العناصر يمكن استخدام حلقات التكرار بالشكل التالي

كود C#
```
for (int i = 0; i < 5; i++)
    Console.WriteLine(intarray[i]);
```

كود VB.NET
```
For i As Integer = 0 To 4
    Console.WriteLine(intarray(i))
Next
```

6. 2. تكوين المصفوفات

الطريقة الاسهل لادخال البيانات إلى المصفوفة بالشكل التالي مثلاً:

كود C#
```
array[0] = 15;
array[1] = 20;
array[2] = 13;
```

كود VB.NET
```
array(0) = 15
array(1) = 20
array(2) = 13
```

أو عن طريق حلقة تكرار ايضاً ، إلا ان هناك طريقة أخرى لادخال البيانات إلى المصفوفة بالشكل التالي مثلاً:

كود C#
```
int[] intarray = new int[] { 15, 20, 13 };
```

كود VB.NET
```
Dim intarray As Integer() = New Integer() {15, 20, 13}
```

6. 3. المصفوفات متعددة الأبعاد

جميع المصفوفات السابقة هي مصفوفات احادية البعد one dimensional ، هناك انواع اخرى من المصفوفات ثنائية او متعددة الابعاد Multi dimensional ، هذا مثال على مصفوفة ثنائية الابعاد - تسمى باسم- Matrix:

كود C#
```
int matrix = new int[3, 3];
```

كود VB.NET
```
Dim matrix As Integer = New Integer(2, 2)
```

سيكون شكل المصفوفة بالشكل التالي (افتراضي) :

0 0 0

0 0 0

0 0 0

ويمكن ادخال البيانات إلى نقطة من المصفوفة بالشكل التالي:

كود C#
```
matrix[1, 2] = 20;
```

كود VB.NET
```
matrix(1, 2) = 20
```

يمكن عمل حلقة تكرار لادخال البيانات ، وليكن عن طريق المستخدم بالشكل التالي مثلاً:

كود C#
```
int matrix = new int[3, 3];
for (int i = 0; i < 3; i++)
    for (int j = 0; j < 3; j++)
        matrix[i, j] = Console.ReadLine();
```

كود VB.NET
```
Dim matrix As Integer = New Integer(2, 2) {}
For i As Integer = 0 To 2
    For j As Integer = 0 To 2
        matrix(i, j) = Console.ReadLine()
    Next
Next
```

والطباعة بنفس الشكل ايضاً.

6 .4. عمل مصفوفة من المصفوفات

يمكن عمل مصفوفة يحتوي كل عنصر منها على مصفوفة بالشكل التالي:

كود C#
```
int[][] complexarray = new int[5][];
```

كود VB.NET
```
Dim complexarray As Integer()() = New Integer(4)()
```

ويمكن الوصول لأي عنصر فيها عن طريق الكود التالي مثلاً:

كود	C#

```
Console.WriteLine(complexarray[1][4]);
```

كود	VB.NET

```
Console.WriteLine(complexarray(1)(4))
```

وهذا ما يعني العنصر رقم 4 من المصفوفة الأولى الموجودة ضمن المصفوفة complexarray

6 .5. ارسال و استقبال المصفوفات من و إلى الدوال

يمكنك عمل دالة لطباعة محتويات مصفوفة بالشكل التالي:

كود	C#

```
static void print(int[] arr)
{
    for (int i = 0; i < arr.Length; i++)
        Console.WriteLine(arr[0]);
}
```

كود	VB.NET

```
Private Sub print(ByVal arr As Integer())
    For i As Integer = 0 To arr.Length - 1
        Console.WriteLine(arr(0))
    Next
End Sub
```

لا تنسى طبعاً انه يتم التعامل معها byref افتراضياً، لذا أي تعديل في الدالة سيؤثر على المصفوفة الأساسية.

في حالة كون ال array هي ما نود اعادته من الدالة return، نكتب الكود بالشكل التالي:

كود	C#

```
static int[] read()
{
    int[] arr = new int[3];
    for (int i = 0; i < 3; i++)
        arr[i] = Console.Read();
    return arr;
}
```

كود	VB.NET

```
Private Function read() As Integer()
    Dim arr As Integer() = New Integer(2) {}
    For i As Integer = 0 To 2
        arr(i) = Console.Read()
    Next
    Return arr
End Function
```

6. 6. خصائص المصفوفات الرئيسية

تحتوي المصفوفات على بعض خصائص ودوال قد تساعدك في العمل عليها، اشهرها واكثرها استخداماً :

الخاصية	الاستخدام
Length	تحدد طول عناصر المصفوفة
Sort	تقوم بترتيب عناصر المصفوفة
Reverse	Reverse تقوم بعكس ترتيب عناصر المصفوفة
ToString	لتحويل المصفوفة إلى متغير نصي
Rank	Rank تحدد عدد الابعاد في المصفوفة

الجدول 7. 1. خصائص الفئة Array

7. ال Enumeration

اختصار ل - enumerations تحتوي على مجموعة من العناصر تمثل حالات وقيم مختلفة داخل البرنامج.

ابسط مثال على ال Enum هو استخدامها في حالة لموديلات السيارات ، لنفترض لدينا ثلاث انواع من السيارات ، Toyota، Nissan و اخيراً FIAT ، لو كنا نقوم بتخزينهم على شكل قيم 0، 1 و 2 في قاعدة البيانات مثلاً، فإننا لا نريد ان نجبر المبرمج على ادخال رقم كل سيارة، بل يمكنه ادخال اسمها والذي سيتم ترجمته لاحقاً إلى الرقم المناظر له.

كود C#

```
enum cars
{
    toyota = 0,
    nissan = 1,
    fiat = 2
}
```

كود VB.NET

```
Enum cars
    toyota = 0
    nissan = 1
    fiat = 2
End Enum
```

يمكننا ايضاً تحديد المساحة التي يتم تخزين فيها عنصر ال Enum لاختصار المساحة ايضاً ، يمكن تحديد النوع byte كمثال على ذلك بالشكل التالي:

كود C#

```
enum cars : byte
{
    toyota = 0,
    nissan = 1,
    fiat = 2
}
```

كود VB.NET

```
Enum cars As Byte
    toyota = 0
    nissan = 1
    fiat = 2
End Enum
```

الآن يمكننا استخدام الاسماء الجديدة في البرمجة بدلاً من الارقام أو الرموز بما يكفل لنا سهولة الاستخدام ، لنفترض مثلاً اننا نريد برمجة دالة لعرض سعر اي سيارة بناء على اسم السيارة.

كود　C#

```
static void printPrice(cars mycar)
{
    if (mycar == cars.fiat)
        Console.WriteLine("20,000");
    else if (mycar == cars.nissan)
        Console.WriteLine("30,000");
    else
        Console.WriteLine("40,000");
}
```

كود　VB.NET

```
Private Sub printPrice(ByVal mycar As cars)
    If mycar = cars.fiat Then
        Console.WriteLine("20,000")
    ElseIf mycar = cars.nissan Then
        Console.WriteLine("30,000")
    Else
        Console.WriteLine("40,000")
    End If
End Sub
```

هكذا نجد اننا نستخدم اسماء واضحة رغم ان القيم الفعلية المخزنة هي من نوع byte.

اوضح ما يمكنك تخيله كفائدة لل Enum هي دوال API الخاصة بالويندوز ، في الواقع انت ترسل بيانات مثل 00 A127X للدوال ، لكن في الواقع تجد نفسك ترسل بعض الأسماء الواضحة مثل Local كدلالة لمتغير ما بدلاً من الرموز المعقدة والتي سيصعب عليك فهمها دون وجود قاموس لترجمة الرموز .

8. التراكيب Structures

ال Structure او Structs هي انواع مخصصة من البيانات يمكنك انشاءها تستطيع حمل اي خصائص على شكل متغيرات او حتى دوال وخلافه ، ابسط مثال على Struct هو مثال السيارة ، لكل سيارة نجد موديل الصنع واسم الماركة ورقم اللوحة مثلاً ، يمكننا كتابة ال Struct بالشكل التالي مثلاً:

كود C#

```
struct Car
{
    public int carNumber;
    public int year;
    public string factory;
};
```

كود VB.NET

```
Structure Car
    Public carNumber As Integer
    Public year As Integer
    Public factory As String
End Structure
```

والآن عندما نريد تعريف نسخة من (سيارة) فإننا نقوم بتعيين خصائصها بالشكل التالي:

كود C#

```
Car ahmedcar = new Car();
ahmedcar.carNumber = 1000;
ahmedcar.factory = "Nissan";
ahmedcar.year = 2007;
```

كود VB.NET

```
Dim ahmedcar As New Car()
ahmedcar.carNumber = 1000
ahmedcar.factory = "Nissan"
ahmedcar.year = 2007
```

يمكن ان يحتوي ال struct على عنصر هو الآخر بدوره struct ، لو افترضنا struct لرخصة السير يحتوي على اسم المستخدم والسنوات المتبقية لانتهاء الرخصة مثلاً ، فسيكون ذلك بالشكل التالي:

كود	C#

```
struct Licence
{
    public string UserName;
    public int yearsToFinish;
}
```

كود	VB.NET

```
Structure Licence
    Public UserName As String
    Public yearsToFinish As Integer
End Structure
```

والآن لو اردنا ان نجعل رخصة السير جزء من خصائص السيارة ، فسيتم ذلك بالشكل التالي:

كود	C#

```
struct Car
{
    public int carNumber;
    public int year;
    public string factory;
    public Licence carLicence;
};
```

كود	VB.NET

```
Structure Car
    Public carNumber As Integer
    Public year As Integer
    Public factory As String
    Public carLicence As Licence
End Structure
```

ولتحديد خصائص اي سيارة سنكتب كود بالشكل التالي:

كود	C#

```
Car ahmedcar = new Car();
ahmedcar.carNumber = 1000;
ahmedcar.factory = "Nissan";
ahmedcar.year = 2007;
ahmedcar.carLicence.UserName = "Ahmed Gamal";
ahmedcar.carLicence.yearsToFinish = 3;
```

VB.NET	كود

```
Dim ahmedcar As New Car()
ahmedcar.carNumber = 1000
ahmedcar.factory = "Nissan"
ahmedcar.year = 2007
ahmedcar.carLicence.UserName = "Ahmed Gamal"
ahmedcar.carLicence.yearsToFinish = 3
```

8. 1. انشاء الدوال داخل ال Struct

يمكننا في داخل اي struct انشاء دالة لتقوم ببعض العمليات على هذا ال struct، ابسط مثال على ذلك لو اردنا عمل دالة renew لتجديد رخصة السير، يمكن في هذه الحالة كتابة الstruct بالشكل التالي:

C#	كود

```
struct Licence
{
    public string UserName;
    public int yearsToFinish;
    public void renew(int periode)
    {
        yearsToFinish += periode;
    }
}
```

VB.NET	كود

```
Structure Licence
    Public UserName As String
    Public yearsToFinish As Integer

    Public Sub renew(ByVal periode As Integer)
        yearsToFinish += periode
    End Sub
End Structure
```

وهكذا يمكننا تجديد فترة الرخصة عن طريق الوصول إلى هذه الدالة مباشرة.

في التطبيقات الفعلية في العادة لا يسمح لك بالوصول إلى العناصر مباشرة إلا عن طريق دالة وذلك لمنع ادخال بيانات مغلوطة مثلاً . يمكن عمل دالة Create لانشاء ال struct مثلاً وبعد انشاء نسخة من ال struct يتم استدعاء هذه الدالة لملئ البيانات حيث لن يسمح لك برؤية باقي المتغيرات.

هناك حل آخر باستخدام Constructor او المشيد ، حيث يمكنك وقت انشاء نسخة من ال struct تمرير البيانات المطلوبة ، يتم كتابة المشيد بالشكل التالي:

C#	كود

```
struct Car
{
    private int carNumber;
    private int year;
    private string factory;
    private Licence carLicence;
    public Car(int p_carNumber, int p_year, string p_factory, Licence
p_carLicence)
    {
        carNumber = p_carNumber;
        factory = p_factory;
        year = p_year;
        carLicence = p_carLicence;
    }
};
```

VB.NET	كود

```
Structure Car
    Private carNumber As Integer
    Private year As Integer
    Private factory As String
    Private carLicence As Licence
    Public Sub New(ByVal p_carNumber As Integer, ByVal p_year As Integer, ByVal
p_factory As String, ByVal p_carLicence As Licence)
        carNumber = p_carNumber
        factory = p_factory
        year = p_year
        carLicence = p_carLicence
    End Sub
End Structure
```

لاحظ اننا حولنا الوصول إلى المتغيرات ليكون private بحيث لا يمكن الوصول له خارج ال struct، معرفات الوصول ستكون احد مواضيعنا حينما ندخل في عالم ال OOP لذا لا تتعجل

ولكن يكفي ان تعلم ان public تجعل قابلية الوصول من اي مكان إلى المتغير او الدالة ، اما private فلا يمكن الوصول لها إلا من داخل ال struct.

9. الأنواع Value و الأنواع Reference

لو عدنا مرة أخرى إلى درس انواع المتغيرات ، سنجد ان معظم انواع البيانات مشتقة من Value Type، معنى ان هذه الأنواع مشتقة من Value Type ان كل متغير منها يمثل نفسه، فلو افترضنا مثالاً بالشكل التالي:

كود	C#

```
int x=5;
int y=x;
y=18;
Console.WriteLine(x);
Console.WriteLine(y);
```

كود	VB.NET

```
Dim x As Integer = 5
Dim y As Integer = x
y = 18
Console.WriteLine(x)
Console.WriteLine(y)
```

سنجد ان كلاً من x و y له قيمة خاصة به ، برغم اننا ساوينا بينهم في البداية ، السبب اننا في جملة y=x فهم المترجم ان ينشأ متغير y يحصل على قيمة x .

نفس هذا النظام ينطبق على المتغيرات الاساسية ، على ال struct ايضاً...

الأنواع من نوع Reference Types مثل ال Class لا ينطبق عليها المثال ، لنفترض مثال السيارة بالشكل التالي:

C# كود

```
class car
{
    int carNumber;
    string carName;
}
```

VB.NET كود

```
Class car
    Private carNumber As Integer
    Private carName As String
End Class
```

والآن سنكتب نفس المثال السابق مع ال int، سيكون المثال بالشكل التالي:

C# كود

```
Car x = new Car();
x.carNumber = 1;
Car y = x;
Console.WriteLine (x == y);
```

VB.NET كود

```
Dim x As New Car()
x.carNumber = 1
Dim y As Car = x
Console.WriteLine(x = y)
```

لو كنت تظن ان المتغير y يحتوي على نسخة من Car يمكنك التعديل عليها فأنت مخطئ، إذ ان المتغير x و y يشيرون لنفس المتغير ، وبالتالي فأي تعديل في احدهما سيتسبب في تعديل للآخر، يمكنك التأكد من ذلك من خلال كود مثل التالي:

C# كود

```
Car x = newCar();
Car y = x;
x.carNumber=1;
y.carNumber =2;
Console.WriteLine (x.carNumber);
Console.ReadKey();
```

كود	VB.NET

```
Dim x As Car = newCar()
Dim y As Car = x
x.carNumber = 1
y.carNumber = 2
Console.WriteLine(x.carNumber)
Console.ReadKey()
```

لعمل نسخة حقيقة من ال Class المسمى Car لا بد من تطبيق Clone، قم بجعل Car Class يطبق ال Interface ICloneable وقم بكتابة دالة ال Clone.

10. المقارنات

لنعد مرة أخرى لنقطة البداية ، حيث يستخدم اغلب المبرمجين المعامل = أو المعامل == في C# للمقارنة، ابسط امثلة ذلك:

كود	C#

```
if (x == y)
{

}
```

كود	VB.NET

```
if x = y Then

End if
```

تكون القيمة المحصلة True في حالة التساوي فيما تكون False في حالة عدم التساوي ، ونقصد بالتساوي هنا التساوي الكامل مثلاً : 5=5 او 7=7 أو "Ahmed"="Ahmed"، ولكن هذا ينطبق فقط على الانواع البسيطة.

ملاحظة
الأنواع البسيطة هي الانوع مثل String , Long , Int وغيرها...

سنواصل شرحنا مع المثال التالي ايضاً:

كود	C#

```
string a = new string(new char[] { 'h', 'e', 'l', 'l', 'o' });
string b = new string(new char[] { 'h', 'e', 'l', 'l', 'o' });
Console.WriteLine(a == b);
```

كود	VB.NET

```
Dim a As New String(New Char() {"h"C, "e"C, "l"C, "l"C, "o"C})
Dim b As New String(New Char() {"h"C, "e"C, "l"C, "l"C, "o"C})
Console.WriteLine(a = b)
```

في هذه الحالة ستظل النتيجة True ايضاً ، ولكن ماذا لو اكملنا الكود بالأسطر التالية:

كود	C#

```
object c = a;
object d = b;
Console.WriteLine(c == d);
```

كود	VB.NET

```
Dim c As Object = a
Dim d As Object = b
Console.WriteLine(c = d)
```

السطر السابق سيخيب ظنك للاسف ، لإن الناتج سيكون False ، الحل في هذه الحالة باستخدام المعامل Equals

كود	C#

```
Console.WriteLine(c.Equals(d));
```

كود	VB.NET

```
Console.WriteLine(c.Equals(d))
```

استخدام آخر للمعامل Equals للمقارنة بين ال Structs مثلاً ، لنفترض المثال التالي:

كود	C#

```
struct car
{
    public int carNumber;
};
```

كود	VB.NET

```
public structure Car
    public carNumber As Integer
End structure
```

والآن، لو قمنا بتعريف المتغيرات التالية:

كود	C#

```
Car x = new Car();
x.carNumber=1;
Car y=x;
Console.WriteLine (x==y);
```

كود	VB.NET

```
Dim x As New Car()
x.carNumber = 1
Dim y As Car = x
Console.WriteLine(x = y)
```

ما هو الناتج الذي تتوقعه، المفترض ان تكون الرسالة المطبوعة هي True ؟؟؟ للاسف هي حتى ليست False لإن هذا سيعطيك خطأ.

الحل الصحيح هو باستخدام المعامل Equals بالطريقة التالية:

كود	C#

```
Console.WriteLine (x.Equals(y));
```

كود	VB.NET

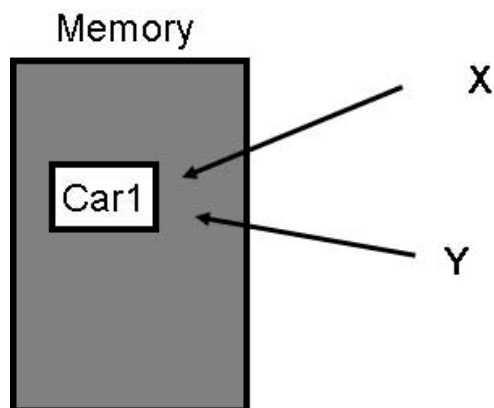
```
Console.WriteLine(x.Equals(y))
```

في حالة تعريف Classes ، يمكن استخدام == ايضاً بالشكل التالي:

كود	C#

```
Console.WriteLine (Class1==Class2);
```

كود	VB.NET

```
Console.WriteLine(Class1 = Class2)
```

أو Equals كما في المثال السابق، هل تعرف لماذا يكون الناتج دائماً True ؟

الاجابة المنطقية لهذا الموضوع هي أنك ستخبرني ان الكائنان يشيران لنفس المكان في الذاكرة

لذا كان الناتج True بالشكل المقابل

وهذا صحيح ، لنفترض المثال التالي:

كود C#

```
Car x = new Car();
Car y = new Car();

x.carNumber=1;
y.carNumber = 1;

Console.WriteLine (x==y);
Console.ReadKey();
```

كود VB.NET

```
Dim x As New Car()
Dim y As New Car()

x.carNumber = 1
y.carNumber = 1

Console.WriteLine(x = y)
Console.ReadKey()
```

ستكون النتيجة هي ... False برغم ان الخصائص الداخلية لهما واحدة ، الحل بكتابة دالة بالشكل التالي:

كود C#

```
bool isthesame(Car c1, Car c2)
{
    if (c1.carNumber == c2.carNumber)
        return true;
    return false;
}
```

كود	VB.NET

```
Private Function isthesame(ByVal c1 As Car, ByVal c2 As Car) As Boolean
    If c1.carNumber = c2.carNumber Then
        Return True
    End If
    Return False
End Function
```

لكن ماذا لو كنا نرغب في عمل نسخة جديدة من المتغير ؟

في هذه الحالة نستخدم مبدأ Clone بالشكل التالي:

كود	C#

```
Car doCopy(Car c1)
{
    Car newCar = new Car();
    newCar.carNumber = c1.carNumber;
    //
    //
    return newCar;
}
```

كود	VB.NET

```
Private Function doCopy(ByVal c1 As Car) As Car
    Dim newCar As New Car()
    newCar.carNumber = c1.carNumber
    '
    '
    Return newCar
End Function
```

في كثير من العناصر الاساسية تجد الدالة Clone موجودة بصورة افتراضية ، اي فئة مشتقة من الواجهة ICloneable ستجد هذه الدالة جاهزة للاستخدام مباشرة، لو كنت انت من تقوم ببرمجة الفئة Class وقمت باشتقاقها من الفئة السابقة فستجد الدالة Clone موجودة لتقوم ببرمجتها بحيث تكون الفئة الخاصة بك على المعايير القياسية ، حيث يستطيع اي مستخدم لفئتك من عمل Clone لها مباشرة.

11. Nullable Types

في عرف المتغيرات التقليدية تعد القيمة null غير مقبولة اطلاقاً، فالمتغير من نوع Boolean مثلاً لا بد ان يحتوي على True أو False ، المتغير Short لا بد ان يحتوي

على ارقام ما بين الصفر و 32767 مثلاً ... وهكذا ، في حين كانت القيم النصية مثلاً String تستطيع استيعاب القيمة null لإنها من نوع reference.

لكننا في بعض الاحيان نضطر لأن يحمل متغير منطقي Boolean قيمة (لا قيمة - null) بمعنى اننا غير قادرين على تحديد فيما إذا كان True أو False ، ابسط امثلة ذلك هي المشكلة التقليدية التي تواجه اغلب مبرمجي قواعد البيانات في حالة عدم ادخال المستخدم لقيمة في حقل ما وليكن حقل (متزوج)، ولنفرض ان المستخدم لم يحدد كون الشخص متزوجاً من عدمه ، في هذه الحالة سيظل الحقل يحتفظ بقيمة null، وهو ما كان يستلزم عمل Check على انها ليست null قبل وضعها في اي متغير من نوع bool مثلاً.

لكن ومع الاصدارة الثانية من net Framework. اصبح بالامكان تعريف متغير مخصص ليسمح بتقبل القيمة null اضافة للقيم الاساسية له ، كما في المثال التالي:

كود C#

```
bool? ismarried = null;
```

كود VB.NET

```
Dim ismarried As System.Nullable(Of Boolean) = Nothing
```

في هذه الحالة يمكننا اسناده لقراءة قيمة مباشرة من قاعدة البيانات دون القلق من كون القيمة الحالية هيnull.

هناك عدة طرق اخرى لتعريف متغير يقبل null مثل الطريقة التالية ايضا:

كود C#

```
Nullable<bool> nullableBool = null;
```

كود VB.NET

```
Dim ismarried As System.Nullable(Of Boolean) = Nothing
```

حيث ان ? هي فعلياً اختصار لل Generic المسمى System.Nullable<T> ، وهو ما سنتعرف على معناه في أجزاء قادمة من الكتاب. وبنفس الطريقة فيما لو اردنا تعريف دالة يكون ال return لها Nullable فسيكون ذلك بالشكل التالي:

كود C#
```csharp
public bool? functionName()
{

}
```

كود VB.NET
```vbnet
Public Function functionName() As System.Nullable(Of Boolean)

End Function
```

11 .1. خصائص ال Nullable

اهم خصائص ال nullable هي خاصية HasValue والتي تحدد فيما إذا كان المتغير به قيمة ام انه يحتفظ بقيمة null بالشكل التالي:

كود C#
```csharp
if (ismarried.HasValue)
{

}
```

كود VB.NET
```vbnet
If ismarried.HasValue Then

End If
```

يمكن كتابتها باستخدام وسائل المقارنة ايضاً != أو <> في الفيجوال بيسك بالشكل التالي:

كود C#
```csharp
if (ismarried != null)
```

كود VB.NET
```vbnet
If ismarried <> Nothing Then
```

ملاحظة

لا تنس أن nothing خاصة بالفيجوال بيسك بدلاً من null في السي شارب

11 .2. المعامل ؟؟

يمكن استخدام المعامل ؟؟ مع القيم ال nullable لكي نخبره بوضع قيمة ما في حالة وجودها ب null، لنفترض مثال حالة الزواج السابق ، وسنفترض ان اي شخص لم يقم بادخال بيانات الزواج فهو شخص اعزب بمعنى ان القيمة ستصبح False مباشرة، سنقوم بكتابة الكود التالي من اجل ذلك:

كود C#

```
int? ismarried = returnvaluefromdatabase() ?? false;
```

كود VB.NET

```
Dim ismarried As System.Nullable(Of Integer) = IIf(returnvaluefromdatabase() Is
Nothing, [False], returnvaluefromdatabase()
```

الباب

8

الفئات ومقدمة إلى البرمجة كائنية التوجه

1. مقدمة إلى الفئات Classes

كما لاحظنا في دروسنا السابقة ، فإن البرنامج المنشأ تحت Console Application يحتوي على فئة واحدة مسماه باسم Class Program تحتوي بداخلها على دوال ومتغيرات واجراءات وطرق ... في الواقع فالبنية الاساسية لأي برنامج يطبق مبادئ OOP هو الفئة Class.

يمكن ان يحتوي البرنامج على عدة فئات ، ويمكن ان تحتوي الفئة ايضاً على عدة فئات بداخله ، في حالة رغبتك في عمل Class منفصل يمكنك اضافة New Class من Project.

يتم تعريف الفئة باستخدام الكلمة المحجوزة class ومن ثم اسمه ، يتم تعريف مكوناته بين {} في السي شارب أو من بعد جملة التعريف حتى End Class في VB.net، سنقوم بادراج فئة جديدة ونسميها باسم Person، سنجد الكود التالي موجوداً بصورة افتراضية:

كود	C#

```
namespace ConsoleApplication4
{
    class Person
    {
    }
}
```

كود	VB.NET

```
Namespace ConsoleApplication4
    Class Person

    End Class
End Namespace
```

الجزء الأول الذي يحدد ال namespace ضروري ، حيث يعني هذا ان جميع الفئات Classes الموجودة تحت نفس ال namespace يمكنها رؤية بعضها الآخر ، لذا ستتمكن من التعامل مع الفئة الجديدة من خلال main الخاصة بنا في الفئة الاساسية المسماه Program.

ابسط مكونات أي فئة Class هي المتغيرات ، يمكننا مثلاً اضافة بيانات الاسم الأول والأخير والعمر داخل الفئة Class بالشكل التالي:

كود C#

```
class Person
{
    public string FirstName;
    public string LastName;
    public int Age;
}
```

كود VB.NET

```
Class Person
    Public FirstName As String
    Public LastName As String
    Public Age As Integer
End Class
```

الآن يمكن تعريف عدة كائنات Objects من هذه الفئة بالشكل التالي مثلاً:

كود C#

```
Person Ahmed = new Person();
Ahmed.Age = 15;
Ahmed.FirstName = "Ahmed";
Ahmed.LastName = "Gamal";
Person Ali = new Person();
Ali.Age = 15;
Ali.FirstName = "Ahmed";
Ali.LastName = "Gamal";
```

كود VB.NET

```
Dim Ahmed As New Person()
Ahmed.Age = 15
Ahmed.FirstName = "Ahmed"
Ahmed.LastName = "Gamal"
Dim Ali As New Person()
Ali.Age = 15
Ali.FirstName = "Ahmed"
Ali.LastName = "Gamal"
```

أو يمكن تعريفهم على شكل مصفوفة بالشكل التالي:

كود C#

```
Person [] MyEmpolyee = new Person[3];
MyEmpolyee[0] = new Person();
MyEmpolyee[0].FirstName = "Ahmed";
MyEmpolyee[0].LastName = "Gamal";
MyEmpolyee[0].Age = 15;
```

كود VB.NET

```
Dim MyEmpolyee As Person() = New Person(2) {}
MyEmpolyee(0) = New Person()
MyEmpolyee(0).FirstName = "Ahmed"
MyEmpolyee(0).LastName = "Gamal"
MyEmpolyee(0).Age = 15
```

كما قلنا في حالة ال struct، فإن بامكاننا ايضاً تعريف الدوال داخل الفئة Class، سنقوم ببرمجة دالة تعيد لنا الاسم الكامل لشخص معين ، ستكون بالشكل التالي:

كود C#

```
public string getFullName()
{
    return FirstName + LastName;
}
```

كود VB.NET

```
Public Function getFullName() As String
    Return FirstName + LastName
End Function
```

1. 1. المشيدات Constructors

عندما نقوم بتعريف new Person() فإن هذا يعني اننا نقوم بتشغيل الدالة ال Constructor، وهي الدالة التي تعمل مع تشغيل اي نسخة من البرنامج ، افتراضياً تكون هذه الدالة خالية ويمكننا وضع بعض الاوامر فيها التي نحتاجها وقت انشاء نسخة ، لعرض مثلاً رسالة تخبرنا بانشاء نسخة جديدة من الفئة Class :

ملاحظة

في السي شارب يتم عمل دالة بنفس اسم الفئة ، اما في فيجوال بيسك فيتم تسمية الدالة باسم New

كود C#
```
public Person()
{
    Console.WriteLine("new object");
}
```

كود VB.NET
```
Public Sub New()
    Console.WriteLine("new object")
End Sub
```

يمكن ايضاً ان يستقبل ال Constructor بارميترس، فمثلاً لجعل الConstructor يستقبل الاسم الأول مع تعريف الاوبجكت الجديد ، فسيكون ذلك بالشكل التالي:

كود C#
```
public Person(string userfirstname)
{
    FirstName = userfirstname;
}
```

كود VB.NET
```
Public Sub New(ByVal userfirstname As String)
    FirstName = userfirstname
End Sub
```

ولعمل نسخة جديدة سيتوجب علينا كتابة الكود التالي:

كود C#
```
Person MyEmpolyee = new Person("ahmed");
```

كود VB.NET
```
Dim MyEmpolyee As New Person("ahmed")
```

1. 2. المهدمات Destructor

عكس ال Constructor، يتم اطلاق هذا الحدث مع انهاء استخدام ال Object، لاصدار صوت Beep مثلاً مع انهاء البرنامج:

كود C#

```
~Person()
{
    Console.Beep();
}
```

كود VB.NET

```
Protected Overrides Sub Finalize()

End Sub
```

نستفيد من ال Destructor في تنفيذ بعض العمليات قبل تدمير الكائن Object تماماً، احياناً ما نحتاج إلى مسح جميع المتغيرات المرتبطة به في الذاكرة وهو الاستخدام الاشهر لهذا الحدث ، ايضاً يمكن استخدامه لتغيير العدادات مثلاً والتي تقوم بعد النسخ من فئة معينة.

2. this

لتوضيح مفهوم الكلمة المحجوزة this في السي شارب، سنعود إلى المثال في الدرس السابق مباشرة ، والذي قمنا فيه بعمل Constructor بالشكل التالي:

كود C#

```
public Person(string userfirstname)
{
    FirstName = userfirstname;
}
```

كود VB.NET

```
Public Sub New(ByVal userfirstname As String)
    FirstName = userfirstname
End Sub
```

لو افترضنا الآن اننا سنقوم بتغيير اسم البارميتر ليكون FirstName بالشكل التالي مثلاً:

كود C#

```
public Person(string FirstName)
{
    FirstName = FirstName;
}
```

كود — VB.NET

```
Public Sub New(ByVal FirstName As String)
    FirstName = FirstName
End Sub
```

في هذه الحالة للاسف سيفهم الكومبايلر انك تجعل FirstName البارميتر يساوي نفسه ، ولن ينظر إلى FirstName كمتغير ضمن الاوبجكت . لتعريف الكومبايلر بإنك تقصد ان المتغير الموجود في الاوبجكت = البارميتر المرسل سنستخدم الكلمة this في C# أو Me في VB.net والتي ستشير إلى المتغير الموجود في الاوبجكت الحالي

كود — C#

```
public Person(string FirstName)
{
    this.FirstName = FirstName;
}
```

كود — VB.NET

```
Public Sub New(ByVal FirstName As String)
    Me.FirstName = FirstName
End Sub
```

حتى في الحالة السابقة التي تطرقنا لها في الدرس السابق ، فإن استخدام this سيكون مناسب دائماً ، ابسط اسباب ذلك هو اظهار قائمة بالعناصر الموجودة في الاوبجكت وقت البرمجة منذ اللحظة التي تكتب فيها this.

3. التعرف على static

لاحظنا في الدالة main الموجودة اساساً في البرنامج اننا نستخدم الكلمة static في ال C#اضافة لمناظرها Shared في ال VB.net ، فما معنى هذه الكلمة ؟

الكلمة static والتي يمكن استخدامها مع الدوال او الفئات او حتى المتغيرات تعني ان هذا المتغير او الدالة يتم الوصول لها من على مستوى ال Class مباشرة وليس من مستوى ال object، لنفترض المثال التالي الخاص ب Person

كود C#

```
class Person
{
    public string FirstName;
    public string LastName;
    public int Age;
    static public int counter;
}
```

كود VB.NET

```
Class Person
  Public FirstName As String
  Public LastName As String
  Public Age As Integer
  Public Shared counter As Integer
End Class
```

ولنفترض اننا كتبنا الكود التالي مثلاً:

كود C#

```
Person MyEmpolyee = new Person();
MyEmpolyee.Age = 15;
MyEmpolyee.counter = 3;
```

كود VB.NET

```
Dim MyEmpolyee As New Person()
MyEmpolyee.Age = 15
MyEmpolyee.counter = 3
```

للاسف لن يكون هذا الكود صحيحاً ، حيث ان counter هو متغير static ولذا تتم قراءته مباشرة على مستوى الفئة Class بالشكل التالي:

كود C#

```
Person MyEmpolyee = new Person();
MyEmpolyee.Age = 15;
Person.counter = 3;
```

كود VB.NET

```
Dim MyEmpolyee As New Person()
MyEmpolyee.Age = 15;
Person.counter = 3;
```

لو قمنا بتعريف Object جديد من نفس الفئة ، وقمنا بعرض قيمة ال counter الخاصة به من داخل الفئة Class سنجد انها 3 مباشرة ، ذلك ان المتغير static يتم تعريف نسخة واحدة منه على مستوى الفئة Class لجميع الكائنات objects التي يتم عملها منه.

مثال على الاستفادة من هذه الخاصية هي خاصية العدادات، والتي تقوم بعد الاوبجكتس التي ننشأها من هذا الفئة Class، سنضع الكود الاضفة في ال Constructor فيما نضع كود النقص في Destructor:

كود C#

```
class Person
{
    public string FirstName;
    public string LastName;
    public int Age;
    static public int counter;
    public string getFullName()
    {
        return FirstName + LastName;
    }
    public Person()
    {
        counter++;
    }
    ~Person()
    {
        counter--;
    }
}
```

كود | VB.NET

```
Class Person
  Public FirstName As String
  Public LastName As String
  Public Age As Integer

  Public Shared counter As Integer

  Public Function getFullName() As String
      Return FirstName + LastName
  End Function

  Public Sub New()
      counter += 1
  End Sub
  Protected Overrides Sub Finalize()
      Try

          counter -= 1
      Finally
          MyBase.Finalize()
      End Try
  End Sub
End Class
```

والآن سنقوم بانشاء عدد من الكائنات ، ومن ثم نقوم بعرض قيمة العداد:

كود | C#

```
Person MyEmpolyee = new Person();
Person MyEmpolyee2 = new Person();
Person MyEmpolyee3 = new Person();
Person MyEmpolyee4 = new Person();
Person MyEmpolyee5 = new Person();
Console.WriteLine(Person.counter);
```

كود | VB.NET

```
Dim MyEmpolyee As New Person()
Dim MyEmpolyee2 As New Person()
Dim MyEmpolyee3 As New Person()
Dim MyEmpolyee4 As New Person()
Dim MyEmpolyee5 As New Person()
Console.WriteLine(Person.counter)
```

والآن لو افترضنا الكود التالي:

كود | C#

```
Person[] MyEmpolyee = new Person[5];
Console.WriteLine(Person.counter);
```

كود | VB.NET

```
Dim MyEmpolyee As Person() = New Person(4) {}
Console.WriteLine(Person.counter)
```

لا تتوقع ان الناتج سيكون 5 ، للاسف توقعك ليس في محله ، والسبب ان اياً من نسخ MyEmpolyee لم يتم انشاءه بعد، الحالة التالية هي من ستعطيك الناتج 5:

كود | C#

```
Person[] MyEmpolyee = new Person[5];
for (int i = 0; i < 5; i++)
    MyEmpolyee[0] = new Person();
Console.WriteLine(Person.counter);
```

كود | VB.NET

```
Dim MyEmpolyee As Person() = New Person(4) {}
For i As Integer = 0 To 4
  MyEmpolyee(0) = New Person()
Next
Console.WriteLine(Person.counter)
```

3. 1. Static Class

منذ net Framewrok 2.0. تم اعتماد مبدأ ال Static Class، وبطبيعة الحال فإن أي Static Class لا بد ان تكون جميع المتغيرات والدوال فيه من نوع static، اضف إلى ذلك فإنه ليس من المنطقي تعريف object من هذا الفئة Class باستخدامnew.

في هذه الحالة يمكن استخدام الفئة Class ودوالة مباشرة ، ابسط امثلة هذا الClass هي MessageBox والتي يمكن استخدامها مباشرة دون تعريف نسخة منها.

كود | C#

```
MessageBox.Show("Welcome");
```

كود	VB.NET

```
MessageBox.Show("Welcome")
```

4. اعادة التحميل OverLoading

من اهم مبادئ ال OOP هي تطبيق مبدئ ال Overloading ، حيث يمكن تسمية عدة طرق او دوال بنفس الاسم ما دام هناك تغيير في البارميترات.

ابسط مثال على عمليات ال Overloading هي دالة MessageBox.Show() في ال Windows Forms

لو جربت ستجد هذه الأوامر مثلاً:

كود	C#

```
MessageBox.Show(string text);
MessageBox.Show(string text,string caption);
MessageBox.Show(string text, string caption, MessageBoxButtons buttons);
```

وغيرها ايضاً ضمن 7 اوامر ، كل واحدة منها عبارة عن دالة خاصة ولهم جميعاً نفس الاسم ، لنتصور لو لم يكن مبدأ Overloading موجوداً ، كنا سنضطر لكتابة دوال بالشكل التالي:

كود	C#

```
MessageBox.ShowWithString(string text);
MessageBox.ShowStringAndCaption(string text,string caption);
MessageBox.ShowStringAndCaptionAndButtons(string text, string caption,
MessageBoxButtons buttons);
```

أما الآن ولتسهيل الاستخدام تجدهم جميعاً بنفس الاسم، وهذا ما يسبب سهولة الاستخدام.

لتطبيق مبدأ ال OverLoading في دوالك يمكنك كتابة كود بالشكل التالي - لعملية الجمع مثلاً في حين انك لا تدري هل يمرر لك في الفئة class قيم من نوع int او من نوع double

كود　C#

```
static int Add(int x, int y)
{
    return x + y;
}
static double Add(double x, double y)
{
    return x + y;
}
static long Add(long x, long y)
{
    return x + y;
}
```

كود　VB.NET

```
Private Shared Function Add(ByVal x As Integer, ByVal y As Integer) As Integer
    Return x + y
End Function
Private Shared Function Add(ByVal x As Double, ByVal y As Double) As Double
    Return x + y
End Function
Private Shared Function Add(ByVal x As Long, ByVal y As Long) As Long
    Return x + y
End Function
```

يمكن ان نعيد الكرة مرة أخرى مع ال Constructor ، فيمكنك عمل عدد منها حسب احتياج المستخدم ، لو عدنا لمثالنا Person يمكن تعريف ال Constructors بالشكل التالي:

كود　C#

```
class Person
{
    public string FirstName;
    public string LastName;
    public int Age;
    public Person(){
        Console.WriteLine("c1");
    }
    public Person(string fname, string lname)
    {
        FirstName = fname;
        LastName = lname;
        Console.WriteLine("c2");
    }
    public Person(string fname, string lname, int myage)
    {
        FirstName = fname;
        LastName = lname;
        Age = myage;
        Console.WriteLine("c3");
    }
}
```

كود VB.NET

```
Class Person
    Public FirstName As String
    Public LastName As String
    Public Age As Integer

    Public Sub New()
        Console.WriteLine("c1")
    End Sub

    Public Sub New(ByVal fname As String, ByVal lname As String)
        FirstName = fname
        LastName = lname
        Console.WriteLine("c2")
    End Sub

    Public Sub New(ByVal fname As String, ByVal lname As String, ByVal myage As
Integer)
        FirstName = fname
        LastName = lname
        Age = myage
        Console.WriteLine("c3")
    End Sub
End Class
```

وبهذه الطريقة يمكننا انشاء كائنات بأحد الطرق التالية:

كود C#

```
Person MyEmpolyee = new Person();
Person MyEmpolyee = new Person("ahmed", "gamal");
Person MyEmpolyee = new Person("ahmed", "gamal", 22);
```

كود VB.NET

```
Dim MyEmpolyee As New Person()
Dim MyEmpolyee As New Person("ahmed", "gamal")
Dim MyEmpolyee As New Person("ahmed", "gamal", 22)
```

5. معرفات الوصول Access Modifiers

في دروسنا السابقة ، كان معرف الوصول الافتراضي الذي نقوم باستخدامه هو public، وذكرنا ان هذا يتيح لنا الوصول إلى هذه الدالة أو هذا المتغير مباشرة من خلال ال Object، قلنا ايضاً إننا لو استخدمنا المعرف private فلن نتمكن من رؤية هذا المتغير او هذه الدالة سوى من داخل الفئة class نفسه فقط

في هذا الدرس سوف نتعرف على انواع معرفات الوصول المختلفة والفروقات بينها.

:Public

يتيح لك هذا المعرف الوصول إلى الدالة أو المتغير من ال Object مباشرة ، ايضاً لو قام أحد باشتقاق الفئة Class فسيجد هذه الدالة public كما هي وسيمكن لل objects من الفئة Class الجديد الوصول لها.

:Private

يمكن فقط رؤيته داخل الفئة Class او ال Struct الذي تم تعريفه من خلاله.

:Protected

لا يمكن الوصول المباشر لها من ال Object ولكن يمكن الوصول لها من الفئة Class المشتق.

:internal

يمكن الوصول لها داخل الاسمبلي الحالي فقط.

:protected internal

يمكن الوصول لها ايضاً من خلال الفئة Class المشتق اضافة للوصول لها من خلال نفس ملف الاسمبلي.

6. العناصر الأساسية في ال OOP

جميع لغات البرمجة التي تطبق مبادئ ال OOP لا بد ان تتعرض للنقاط الثلاث التالية:

1. Encapsulation
2. Inheritance
3. Polymorphism

6. 1. Encapsulation

يقصد بال Encapsulation هو اخفاء وضم البيانات والأكواد المختلفة واظهارها في النهاية للمستخدم على شكل Black Box ، حيث ستجد في النهاية دالة باسم PrintReport ، في الواقع لا يهمك ما هو الكود الموجود داخلها ، حيث يكفيك استدعاءها لتعمل لديك دالة طباعة التقارير.

تستفيد ايضاً من هذه الخاصية في حالة العمل الجماعي على المشروع ، في الحالة العمل باسلوب Structured ستجد جميع الكود في مكان واحد ، اما مع ال Encapsulation في ال OOP فسيكون كل جزء من العمل مغلقاً على نفسه ويتم التواصل بينهم البعض عن طريق Objects مختلفة ، وهذا ما يساهم في تسهيل تركيب العمل ، تسهيل اكتشاف الأخطاء.

6. 2. Inheritance

أو الوراثة ، ويقصد بها عمل نسخة جديدة من الفئة class تحتوي على نفس خصائصه من اجل تطويرها أو تعديل بعض الخصائص ، لنفترض مثلاً ان لدينا فئة class (سيارة) يحتوي على اسم السيارة وتاريخ صنعها مثلاً.

والآن نريد القيام بعمل فئة class للسيارات ال BMW مثلاً ، في هذه الحالة من الخطأ ان نقوم بعمل فئة class منفصل لها ، ولكننا نستطيع اشتقاق فئة class جديد من الفئة class المسمى Car مع اضافة خاصية isSport للفئة class المسماه BMW ، الهدف من ذلك اننا

سنجد خصائص مشتركة بين الCar و ال BMW ، ايضاً سنضطر يوماً لاضافة فئة لل GMC والذي سيحتوي بدوره على بيانات مشتركة.
السبب الآخر لو قمنا باضافة عنصر جديد مثل countofdoors لكل السيارات ، فهل سنقوم بالتعديل في جميع الفئات ، في حالة قمنا بعمل اشتقاق من الفئة Car فإننا نستطيع التعديل فيه فقط لتطبيق التعديلات.

ربما لا تظهر الأهمية في الفئات الصغيرة ولكن تستطيع تخيل حجم الفائدة في حالة وجود كم كبير من البيانات.

يتم تعريف العلاقة Inheritance باسم is a ، حيث ان BMW is a Car ، ايضاً في حالة وجود فئة للاشكال وفئة أخرى للدائرة مثلاً فإن Circle is a Shape .

هناك نوع آخر من العلاقات بين ال Classes وهي علاقة has a وتسمى باسم Aggregation، في هذه الحالة تكون الفئة الثانية محتوية على الفئة الأولى ، مثلاً لو كان لدينا فئة من اجل عجلات السيارة باسم Wheel، ستكون Wheel has a Car .

6 .3 Polymorphism

المبدأ الأخير من مبادئ OOP هو السماح بكتابة فئة بدوالها بدون أي Implementation، والسبب هو اجبار اي مستخدم يقوم باشتقاق الفئة بعمل implementation لهذه الدوال بما يضمن تشابه الاسماء ، ابسط مثال على ذلك فئة Class لـ Shape ، حيث يمكننا تعريفه بالشكل التالي:

كود	C#

```
class Shape
{
    public void Draw()
    {
    }
    public void GetSize()
    {
    }
}
```

كود	VB.NET

```
Class Shape
    Public Sub Draw()

    End Sub
    Public Sub GetSize()

    End Sub
End Class
```

في هذه الحالة يعرف ال Class باسم Abstract Class، توفر C# و VB.net ايضاً مبدأ ال Interface والذي يشبه كثيراً ال Abstract Class ولكن مع فروقات سنتعرف عليها في حينها.
في الفئة السابق فإن كل فئة (دائرة ، مربع ، مثلث) سيتم اشتقاقه من هذا الفئة سيكون مجبراً على كتابة اكواد الرسم وحساب المساحة لكل منهم .
يمكن عمل ذلك بطريقة أخرى عبر ال OverRidding ولكن مع الاختلاف في احتمالية وجود Default Implementation.

7. Encapsulation

يهدف مبدأ ال Encapsulation والذي سبق شرحه باختصار إلى اخفاء التفاصيل الداخلية لأي فئة عن اعين باقي المبرمجين ، باختصار شديد لنفترض نظام محاسبي يتضمن نظام لادارة المخازن ، وآخر للصيانة ، وثالث من اجل المبيعات.

في هذه الحالة يدعوك مبدأ ال Encapsulation ليكون لكل واحد من هذه النظم عدد محدد من الدوال للدخول والخروج من هذا النظام والتي يمكن للنظم الثلاثة التواصل من خلالها ، فمثلاً في نظكام المبيعات تجد (اضافة عملية مبيعات) (اضافة مشتريات) (خصم) ... الخ في مجموعة محدودة جداً من الدوال.

طبعاً لو لاحظت ان عملية مثل اضافة عملية مبيعات تتطلب طابوراً من الأوامر ، يتضمن فتح قاعدة البيانات والتأكد من ان البيانات المدخلة صحيحة والتأكد من وجود الكمية ومن ثم تخزين الناتج في قاعدة البيانات ، تم تجميعها في النهاية على شكل أمر واحد يقوم زميلك الآخر الذي يقوم ببرمجة واجهات المستخدم إلى استخدامه بدلاً من الغوص في كل هذه التفاصيل الفنية.

كما لاحظت ، يفيد هذا الموضوع الأشخاص التي تعمل في مجموعات اولاً ، حيث لن اكون مضطراً لفهم كودك بالكامل ويكفيني ان اعرف كيف اتعامل معك ، كما ان الكود سيكون مصمماً على شكل هرم حيث كل أمر يستتبعه مجموعة من الأوامر ، لكن سيكون زميلك المبرمج قادراً فقط للوصول إلى رأس الهرم وهو ما يقلل كثيراً من الأخطاء ، كما يحمي متغيراتك الخاصة من العبث بها عن طريق الخطأ من المبرمجين الآخرين.

هذا المفهوم الذي تحاول ان توصله لك كل لغات برمجة OOP، ولتطبيق هذا المفهوم هناك العديد من النقاط التي سنتطرق لها في هذا الدرس.

قبل ان نواصل ، اول نقطة لا بد ان تضعها في ذهنك انك مبرمج وهناك مبرمج آخر هو المستخدم ... بمعنى مثلاً انك تقوم ببرمجة فئة ويقوم زميلك المبرمج باستخدام هذ الفئة كجزء من المشروع.

7. 1. استخدام دوال public للوصول إلى متغيرات private

لنفترض المثال التالي بخصوص ال Person الذي قمنا بانشاءه عدة مرات على مستوى الدروس السابقة:

كود C#

```
class Person
{
    public int Age;
}
```

كود VB.NET

```
Class Person

    Public Age As Integer

End Class
```

ولإن زميلك العزيز في الفريق الخاص بادخال بعض البيانات بشر - وجل من لا يسهو - قام بكتابة الجملة التالية:

كود C#

```
Person Ahmed = new Person();
Ahmed.Age = 999999999999;
```

كود	VB.NET

```
Dim Ahmed As New Person()
Ahmed.Age = 999999999999
```

ولو ان السهو في رقم بهذا الحجم يعد شبه مستحيل ، لكننا قد نتعرض لهذه المشكلة في اشياء اقرب للواقع ، وابسط من ذلك لو انه يقوم بقراءة القيمة من المستخدم ليرسلها لك مباشرة ، وفي وجود مستخدم - غلس - فإن مثل هذه المدخلات تعد أمراً طبيعياً.

طبعاً هذا المدخل سيسبب الدمار العاجل للفئة الذي قمت بعملها ، كما ان زميلك مبرمج شاشات الادخال ربما لن يكلف نفسه عناء برمجة امر التحقق من الادخال ، لذا تأتي النصيحة الدائمة ، امنع متغيراتك من الظهور لمستخدمها - المبرمج الآخر - وضع بدلاً منها دوال لقراءتها او الكتابة إليها بالشكل التالي:

كود	C#

```
class person
{
    private int Age;
    public string SetAge(int x)
    {
        if (x > 100 || x < 1)
            return "you can't edit age like that";

        Age = x;
        return "done";

    }
    public int GetAge()
    {
        return Age;
    }
}
```

كود	VB.NET

```
Class person
    Private Age As Integer

    Public Function SetAge(ByVal x As Integer) As String
        If x > 100 OrElse x < 1 Then
            Return "you can't edit age like that"
        End If
        Age = x
        Return "done"
    End Function

    Public Function GetAge() As Integer
        Return Age
    End Function
End Class
```

طبعاً تسمية المتغيرات باسم X هو خطا كما اتفقنا سابقاً ، دالة Set ينبغي ان تأخذ الشكل التالي مثلاً:

كود	C#

```
public string SetAge(int Age)
{
    if (Age > 100 || Age < 1)
        return "you can't edit age like that";

    this.Age = Age;
    return "done";
}
```

كود	VB.NET

```
Public Function SetAge(ByVal Age As Integer) As String
    If Age > 100 OrElse Age < 1 Then
        Return "you can't edit age like that"
    End If

    Me.Age = Age
    Return "done"
End Function
```

وذلك حسبما تعلمنا من خصائص this أو Me في الدرس السابق...

ربما يبدو لك الأمر مملاً او طويلاً ، لكنه مثل هذه الأمور في المشاريع الجدية تعد نقاطاً حيوية لا يمكن الاستغناء عنها لإنها قد تتسبب في سقوط للمشروع يكلف عدة ملايين بسبب خطأ بسيط.

ملاحظة

الدالة GetAge يطلق عليها اسم Accessor أما الدالة SetAge فتسمى باسم Mutator

7. 2. ارسال القيم كجزء من ال Constructor

طريقة أخرى يمكنك استخدامها كبديل او مع الطريقة السابقة ، وهي اتاحة الفرصة للمستخدم لتمرير بارميترات قيم المتغيرات ضمن المشيد Constructor، في الفئات الكبيرة سيكون لزاماً عليك استخدام الطريقة الأولى إلى جانب هذه الطريقة حيث انك لن تتصور المستخدم يقوم بتمرير قيم كل المتغيرات لحظة انشاء Object من ال Class

مثال هذه الطريقة:

كود C#

```
class Person
{
    private int Age;
    public Person(int Age)
    {
        if (Age > 100 || Age < 1)
            return "you can't edit age like that";

        this.Age = Age;
        return "done";
    }
}
```

كود VB.NET

```
Class Person

    Private Age As Integer

    Public Sub New(ByVal Age As Integer)

        If Age > 100 OrElse Age < 1 Then
            Return "you can't edit age like that"
        End If

        Me.Age = Age
        Return "done"
    End Sub
End Class
```

7. 3. استخدام ال Type Property

بنفس الطريقة السابقة ، ولكن بطريقة اخرى واعتماداً على تحويل المتغير إلى خاصية لكل منها دالتان Set و Get ، يتم كتابة ذلك بالشكل التالي:

كود — C#

```
class Person
{
    public int Age
    {
        get { return Age; }
        set
        {
            if (!(value > 100 || value < 1))
                Age = value;
        }
    }
}
```

كود — VB.NET

```
Class Person
    Public Property Age() As Integer
        Get
            Return Age
        End Get
        Set(ByVal value As Integer)
            If Not (value > 100 OrElse value < 1) Then
                Age = value
            End If
        End Set
    End Property
End Class
```

والآن يمكنك الوصول المباشر إلى Age ، ولكن قبل تطبيق اي شيء سيتم استدعاء Set ، وفي حالة طلب شيء سيتم الحصول عليه من Get ، وهو ما يتيح لك التأكد من بيانات الادخال او عمل بعض العمليات على عمليات الاخراج.

لعلك لاحظت ايضاً أن Set و Get هما public لان الخاصية (وليس المتغير في هذه الحالة المسماه Age هي public ، ولكن منذ net 2.0. اصبح بامكانك التعديل في معرف الوصول الخاص ب Set او Get على حدة ، بالشكل التالي مثلاً:

كود C#

```
public int Age
{
    get { return Age; }
    protected set { Age = value; }
}
```

كود VB.NET

```
Public Property Age() As Integer
    Get
        Return Age
    End Get
    Protected Set(ByVal value As Integer)
        Age = value
    End Set
End Property
```

إذا كنت ترغب في عمل Read Only Property فيمكنك ازالة الخاصية Set، أما لو اردت العكس لعمل Write Only Property فأزل الخاصية Get، هذا المثال لRead Only Property :

كود C#

```
public int Age
{
    get { return Age; }
}
```

كود VB.NET

```
Public ReadOnly Property Age() As Integer
    Get
        Return Age
    End Get
End Property
```

8. الوراثة Inheritance

ذكرنا في درس سابق ان OOP لها ثلاث عناصر اساسية ، قمنا بشرح المفهوم الأول Encapsulation في دروس سابقة ، والآن موعدنا مع ال Inheritance .

الآن سنبدأ بعمل مثال نتابع معه العمل ، لنفترض المثال السابق الذي شرحناه الخاص بالعربة:

كود C#

```
class Car
{
    private string carName;
    private int carModel;
    public Car(string carName, int carNumber)
    {
        this.carNme = carName;
        this.carNumber = carNumber;
    }
    public Car()
    {
        carName = "Unknown";
        carNumber = 0;
    }
}
```

كود VB.NET

```
Class Car

    Private carName As String
    Private carModel As Integer
    Public Sub New(ByVal carName As String, ByVal carNumber As Integer)
        Me.carNme = carName
        Me.carNumber = carNumber
    End Sub
    Public Sub New()
        carName = "Unknown"
        carNumber = 0
    End Sub

End Class
```

سنبدأ بهذا المثال البسيط ، ونتابع العمل على تطويره وتحسينه خلال مراحل هذا الدرس.

8. 1. تعريف العلاقة is-a

كما ذكرنا في الدرس السابع عشر ، فإن العلاقة قد تكون is-a وقد تكون has-a ، سنحاول الآن شرح النوع الأول من العلاقات والذي يعني ان الفئة المشتقة هو من نوع الفئة الرئيسية ، سنفترض سيارة BMW

كود C#

```
class BMW : Car
{
}
```

كود	VB.NET

```
Class BMW
    Inherits Car

End Class
```

هكذا نستطيع ان نقول ان الفئة البنت BMW يحتوي على نفس خصائص الفئة الأم Car ونفس دواله وطرقه ال public فقط ، ولكن لنفترض اننا في الفئة الابن نحاول الوصول المباشر إلى الخاصية carName فلن نتمكن من ذلك ، هذا المثال يوضح هذه النقطة:

كود	C#

```
BMW ahmedcar = new BMW();
ahmedcar.carName = "anyname";
```

كود	VB.NET

```
Dim ahmedcar As New BMW()
ahmedcar.carName = "anyname"
```

حتى لو قمنا بتعريف بعض الدوال داخل الفئة المشتقة بحيث تستطيع الوصول إلى هذه الخاصية ، لنفترض اننا اعدنا صياغة الفئة BMW ليكون بالشكل التالي:

كود	C#

```
class BMW: Car
{
    public void changeCarName (string value)
    {
        carName = value;
    }
}
```

كود	VB.NET

```
Class BMW
    Inherits Car
    Public Sub changeCarName (ByVal value As String)
        carName = value
    End Sub
End Class
```

للاسف لن يكون هذا صحيحاً تماماً ، حيث انك بالرجوع إلى درس معرفات الوصول ستكشتف ان معرف الوصول private لا يمكن الوصول له من الفئة المشتقة، من اجل هذا نستخدم معرف

الوصول protected حيث انه يشبه ال private في كونه لا يمكن الوصول المباشر له من خلال ال object ، لكنه في المقابل يمكن الوصول إليه من داخل الفئة المشتقة ، لو افترضنا مثال الفئة Car بالشكل التالي :

كود — C#
```
class Car
{
    protected string carName;
    protected int carModel;
}
```

كود — VB.NET
```
Class Car
    Protected carName As String
    Protected carModel As Integer
End Class
```

في هذه الحالة يمكننا تعريف دالة داخل الفئة المشتقة BMW تقوم بقراءة هذه المتغيرات ، لذا سوف يكون الكود التالي صحيحاً:

كود — C#
```
class BMW: Car
{
    public void changeCarName(string value)
    {
        carName = value;
    }
}
```

كود — VB.NET
```
Class BMW
    Inherits Car

    Public Sub changeCarName(ByVal value As String)
        carName = value
    End Sub
End Class
```

8 .2. الكلمة المحجوزة NotInheritable - sealed

يعني استخدام هذه الكلمة ان هذا الفئة لا يمكن الاشتقاق منه ، يتم ذلك بالشكل التالي:

كود C#
```
sealed class Car
{

}
```

كود VB.NET
```
NotInheritable Class Car

End Class
```

ملاحظة

الكلمة NotInheritable تقابل sealed في C#

8 .3. الوراثة المتعددة

لا توفر لغة السي شارب او ال VB.net مبدأ الوراثة المتعددة ، في حين تطبقه فقط managed c++، معنى كلمة الوراثة المتعددة ان بامكان فئة ما ان تشتق من اكثر من فئة ، لنفترض لدينا فئة شاحنة وفئة سيارة ، في حالة دعم لغة ما للوراثة المتعددة فإننا نستطيع عمل نوع جديد يحتوي على خصائص الشاحنة والسيارة العادية ، ولكن هذا ما لا توفره كل من السي شارب او ال VB.net .

وكبديل لذلك ، تقدم اللغتان دعم لعمل Implementation لاكثر من interface ، وهو ما سنتعرف عليه حينما نصل إلى هذا الجزء.

أما لماذا لم تقدم مايكروسوفت دعم الوراثة المتعددة في C# و VB.net ، إليك هذا الرابط:

رابط

http://blogs.msdn.com/csharpfaq/archive/2004/03/07/85562.aspx

8. 4. التعديل في الفئات المشتقة

كما رأينا في الدروس السابقة ، يمكننا التعديل مباشرة على خصائص وطرق وأحداث ودوال الفئة الجديدة ، في المثال السابق قمنا باضافة دالة تسمح لنا بتغيير الاسم ، يمكننا اضافة خاصية جديدة للفئة BMW تحتوي على عدد أجهزة التلفزيون داخل السيارة كميزة اضافية في السيارات من نوع BMW، وهكذا...

يمكننا عمل فئة أخرى لعربة فيراري ، في هذه الحالة يمكننا اضافة عدد الفتحات الجانبية للمحرك . لكن لو احتجنا في مرحلة الى تعريف خاصية maxSpeed لجميع السيارات فيكفي اضافتها في الفئة الاساسية Car وستجدها موجودة تلقائياً في الفئات الأبناء جميعاً.

من هنا نستطيع ان نلاحظ ان واحدة من الفوائد الرئيسية لعملية الوراثة هي وضع قاعدة عامة للعناصر المتشابهة ، وعمل نسخ لاضافة نقاط الاختلاف فقط بدلاً من اعادة تكرار كل منها عدة مرات ، ربما لن تجد الفرق كبيراً في المثال السابق حيث اننا نعمل مع 3 او 4 خصائص فقط ، ولكن في مثال حقيقي مع عدة اوامر للتعامل مع المستخدم وللحفظ في قاعدة البيانات والطباعة والعرض والعمليات الحسابية ستستطيع ان تدرك الفارق بين استخدام مفهوم الوراثة وعدمه.

8. 5. العلاقة من نوع has-a

كما اوضحنا في اول درسنا فهذا هو النوع الثاني من العلاقات بين الفئات المختلفة، هذا النوع يعني ان الفئة تحتوي على فئة أخرى ، لو افترضنا مثال فئة العجلات بالشكل التالي:

كود | C#

```
class Tires
{
    int TiresType;
    int TiresSize;
}
```

كود VB.NET

```
Class Tires
    Private TiresType As Integer
    Private TiresSize As Integer
End Class
```

نعرف يقينا ان الاطارات ليست من نوع سيارة Car is not a Tire ولكنها جزء من السيارة Car has a Tire، لذا يمكننا تعريف فئة السيارة بالشكل التالي :

كود C#

```
class Car
{
    Tires carTires = new Tires();
}
```

كود VB.NET

```
Class Car
    Private carTires As New Tires()
End Class
```

ملاحظة

لا تنس ان بامكانك تعريفها كـ private او protected وعمل خاصية لها من اجل القراءة والكتابة إليها

8. 6. التحويلات Casting

يقصد بال Casting عموماً هو التحويل من نوع إلى آخر ، تم شرح المفهوم العام له وانواعه في درس سابق .

والآن سنحاول تطبيق نفس المفاهيم على ال Classes، الطريقة الأولى للتحويل هي استخدام "cast" العادية ، مثلاً لو قمنا بتعريف سيارة BMW:

كود C#

```
BMW ahmedCar = new BMW();
```

كود VB.NET
```
Dim ahmedCar As New BMW()
```

وقمنا بارسال المتغير إلى دالة تقوم باستقبال BMW فسوف تعمل بصورة صحيحة ، ايضاً لو قمنا بارسالها إلى دالة تستقبل Car فسيكون هذا صحيحا لإن كل BMW هي في الحقيقة Car ، بينما العملية العكسية ليست صحيحة .

النقطة الثانية لو قمنا بتعريف BMW بالصورة التالية:

كود C#
```
Car ahmedCar = new BMW();
```

كود VB.NET
```
Dim ahmedCar As Car = New BMW()
```

هذا الموضوع صحيح فعلاً وهو ما يدعى باسم implicit cast ، والآن يمكن ارسال المتغير مباشرة إلى تلك الدالة التي تستقبل Car.

لنفترض مثالاً آخر قمنا فيه بتعريف BMW بالشكل التالي:

كود C#
```
Object ahmedCar = new BMW();
```

كود VB.NET
```
Dim ahmedCar As Object = New BMW()
```

هذا صحيح ايضاً لإن كل فئة هو Object ايضاً ، لكن لو قمنا بارسال المتغير إلى الدالة التي تستقبل Car فسوف تظهر رسالة خطا ، لذا نقوم بعمل cast بأحد الاشكال التالية:

كود C#
```
functionname((Car)ahmedCar);
functionname((BMW)ahmedCar);
```

كود — VB.NET

```
functionname(DirectCast(ahmedCar, Car))
functionname(DirectCast(ahmedCar, BMW))
```

8. 7. الكلمة المحجوزة is

تقوم هذه الكلمة باختبار فيما إذا كان الطرف الاول هو من الطرف الثاني ، مثال

كود — C#

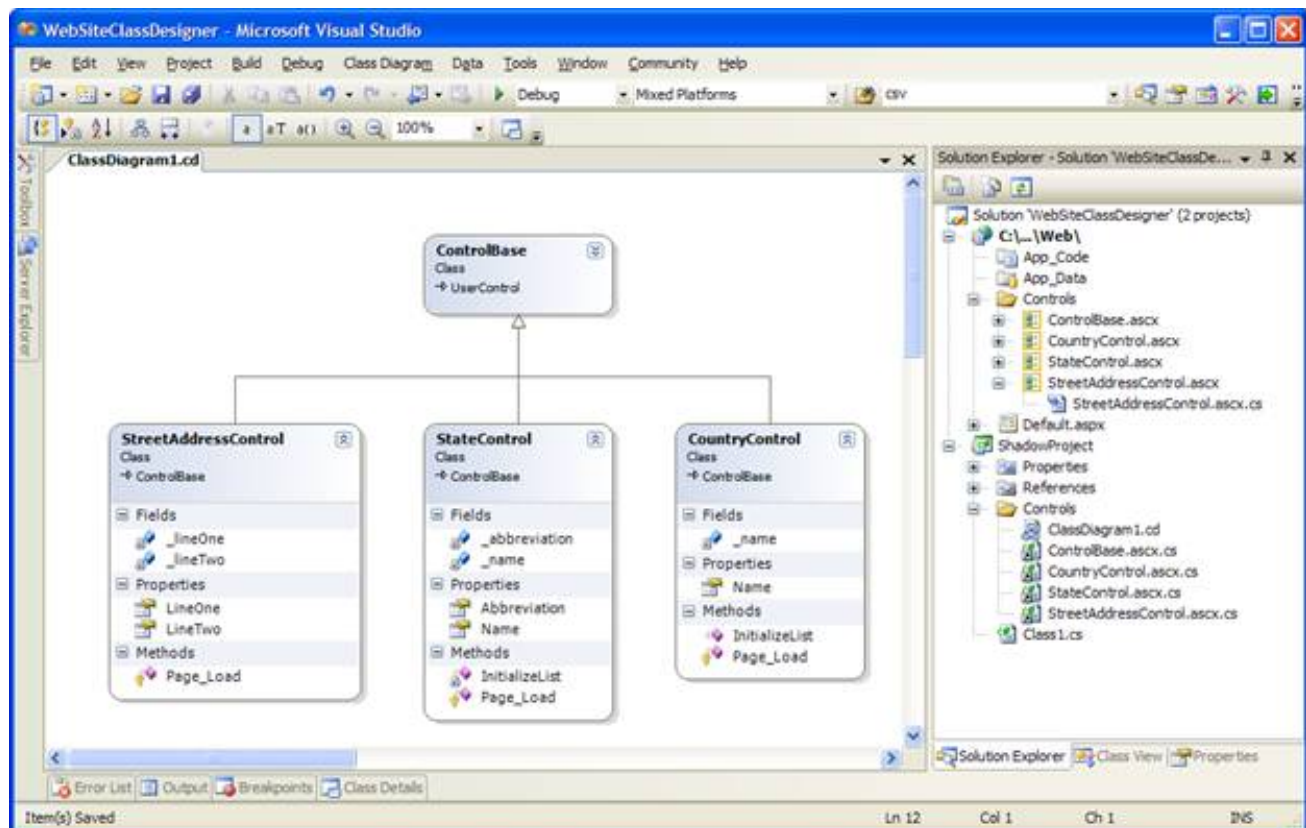
```
if (ahmedCar is BMW)
{

}
```

كود — VB.NET

```
If TypeOf ahmedCar Is BMW Then

End If
```

تفيدك هذه الكلمة في حالة وجود عدة متغيرات من عدة انوع مشتقة من نفس الفئة ، ونريد ان نعرف فيما إذا كانت من نوع BMW او فيراري مثلاً.

8. 8. Visual Studio Class Diagram

يوفر لك الفيجوال ستوديو ابتداء من الاصدار 2005 اداة لعمل ال Class Diagram ، هذا مثال عليها:

الصورة 8. 1. Visual Studio Class Designer

يمكنك اضافتها من new -> class diagram ، ومن ثم العمل عليها مباشرة ، او عرض الفئات التي لديك ، يمكنك انشاء العلاقات المختلفة في هذا الوضع .

9. الـ Polymorphism

تعرفنا في درس سابق على معنى ال polymorphism بشكل عام ، الآن سنتعرف عليه بصورة أكثر تفصيلاً ...

التعريف الماضي والذي يسمح لك بتعريف class بدون اي implementation ، يوفر لك ايضاً كتابة كود ولكن في المقابل يسمح لك بتغيير الimplementation الخاص بالدالة بين الفئة والأخرى ، وهو ما يعرف باسم Overriding.

لنفترض مثال السيارة الخاص بنا ، لو افترضنا وجود الدالة التالية الخاصة بحساب المسافة المتبقية للسيارة قبل ان ينفد البنزين ، وحيث ان السيارة (مثلاً) تصرف جالوناً لكل 10 كيلومتر.

كود — C#

```
public int calc(int fuel)
{
    return fuel * 10;
}
```

كود — VB.NET

```
Public Function calc(ByVal fuel As Integer) As Integer
    Return fuel * 10
End Function
```

ولكن بعد عملنا للسيارة BMW اكتشفنا ان هذه السيارة لا تسير على نفس القاعدة ، إذ انها تصرف جالوناً لكل 5 كيلومترات فقط ، في هذه الحالة نحن بحاجة إلى اعادة تعريف الدالة calc في الفئة المشتقة ، هذا هو ما يعرف باسم اعادة القيادة Overriding وهو احد مبادئ ال polymorphism. يتم ذلك عن طريق تعريف الدالة الاساسية في ال base class من نوع virtual بالشكل التالي:

كود — C#

```
public virtual int calc(int fuel)
{
    return fuel * 10;
}
```

كود — VB.NET

```
Public Overridable Function calc(ByVal fuel As Integer) As Integer
    Return fuel * 10
End Function
```

الvirtual او ال overriddeable تعني ان هذه الدالة يمكن اعادة تعريفها في الفئة المشتقة ، كما يمكن عدم تعريفها ايضاً ، لاعادة تعريفها نكتب الكود التالي في الفئة المشتقة :

كود — C#

```
public override int calc(int fuel)
{
    return fuel * 5;
}
```

كود	VB.NET

```
Public Overloads Overrides Function calc(ByVal fuel As Integer) As Integer

    Return fuel * 5

End Function
```

ايضاً يمكنك ترك ال implementation فارغاً في الفئة الرئيسية ومن ثم قم ببرمجتها في الفئات الأبناء ، تعرف الفئة الفارغ من ال implementation باسم Abstract class وهو غير الinterface الذي سنتعرف عليه في مرحلة قادمة.

10. الـ Abstract

في الدرس السابق تعرفنا على معنى كلمة Abstract class ، وعرفنا انها مجرد فئة عادية بدون implementation، لكن لكي نجبر انفسنا على عمل نسخة منه قبل استخدامه ، نستخدم الكلمة المحجوزة abstract لتعريفة بالشكل التالي:

كود	C#

```
abstract partial class Car
{

}
```

كود	VB.NET

```
Partial MustInherit Class Car

End Class
```

ملاحظة

الكلمة المحجوزة MustInherit في vb.net هي مناظر كلمة abstract في C#

في هذه الحالة سوف يعطينا هذا التعريف خطأ :

كود	C#

```
Car ahmedCar = new Car();
```

VB.NET كود

```
Dim ahmedCar As Car=new Car()
```

10. 1. Abstract Method

ذكرنا في الدرس السابق ايضاً اننا نقوم بتعريف الدوال virtual لكل نقوم بعمل overriding لها في الفئة المشتقة لاحقاً ، وهذا ما يعطينا الحق في اعادة كتابة الكود الخاص بها او تجاهله، لكن لو اردنا اجبار المبرمج على اعادة القيادة overriding نقوم بذلك بتعريف Abstract method بالشكل التالي:

C# كود

```
public abstract int calc(int fuel);
```

VB.NET كود

```
Public MustOverride Function calc(ByVal fuel As Integer) As Integer
```

الباب

9

الواجهات Interfaces

1. تعريف ال Interface

ال interface هو abstract class يحتوي على abstract methods and members يمكن عمل نسخة منه باستخدام بمفهوم implementation بدلاً من inheritance ، كما يتميز بامكانية عمل implement لاكثر من interface في المرة الواحدة وهو البديل عن مفهوم multiple inheritance .

يمكن تعريف interface بالشكل التالي مثلاً:

كود | C#

```
public interface ICar
{
    int carMaxSpeed { get; set; }
    void AddItem(string Item);
}
```

كود | VB.NET

```
Public Interface ICar
    Property carMaxSpeed() As Integer
    Sub AddItem(ByVal Item As String)

End Interface
```

لا تنسى ان بامكانك اشتقاق interface من آخر بالشكل التالي مثلاً:

كود | C#

```
public interface ICar : IDisposable
{
    int carMaxSpeed { get; set; }
    void AddItem(string Item);
}
```

كود | VB.NET

```
Public Interface ICar
    Inherits IDisposable
    Property carMaxSpeed() As Integer
    Sub AddItem(ByVal Item As String)

End Interface
```

ولعمل implements لاي interface نقوم بكتابة الكود التالي:

كود	C#

```
public class BMW : ICar
{

}

public class BMW2 : ICar, ITruck
{

}
```

كود	VB.NET

```
Public Class BMW
    Implements ICar

End Class

Public Class BMW2
    Implements ICar, ITruck

End Class
```

الآن يمكنك اعادة كتابة الدوال الموجودة . اضافة دوال جديدة ، ومن ثم استخدام الفئة كما قمنا بذلك قبلاً.

هذه بعض النقاط التي لا يمكنك عملها مع الواجهات

- بداية ، ال Interface لا يمكن عمل اي Object منه لذا فمثل هذه الجملة خاطئة:

كود	C#

```
ICar x = new ICar();
```

كود	VB.NET

```
Dim x As New ICar()
```

- لا يمكن تعريف متغير عادي في ال Interface، لذا مثل هذه الجملة داخل ال Interface خاطئة:

كود	C#

```
int number;
```

كود	VB.NET

```
Dim number as integer
```

البديل ، هو عمل خاصية set و get بالشكل التالي:

كود	C#

```
int x { set; get; }
```

كود	VB.NET

```
Private Property x() As Integer
    Get

    End Get
    Set(ByVal value As Integer)

    End Set
End Property
```

- بالطبع لا يمكنك كتابة أي سطر كود داخل ال interface.
- كما لا يمكن عمل constructor لل interface.

2. Names Clashes

احياناً ما يقوم class او struct ما بعمل implements لاكثر من interface ، وربما يحدث مشاكل في تشابه الأسماء ، لنفترض المثال التالي:

كود	C#

```
public interface ICar
{
    void move();
}
public interface ITruck
{
    void move();
}
public class BMW : ITruck, ICar
{
}
```

كود	VB.NET

```
Public Interface ICar
    Sub move()
End Interface
Public Interface ITruck
    Sub move()
End Interface
Public Class BMW
    Implements ITruck
    Implements ICar

End Class
```

ربما لن تجد مشكلة فيما لو قمت بعمل implementation للدالة move() في الفئة الجديدة BMW:

كود	C#

```
public class BMW : ITruck, ICar
{
    void move()
    {
        // do something.
    }
}
```

كود	VB.NET

```
Public Class BMW
    Implements ITruck
    Implements ICar

    Private Sub move()
        ' do something.
    End Sub
End Class
```

ولكن ماذا لو أردت عمل implementation للدالة move() من Car واخرى للدالة move() من Truck ، نقوم بذلك بالشكل التالي:

كود C#

```
public class BMW : ITruck, ICar
{
    void ICar.move()
    {
        // do something.
    }
    void ITruck.move()
    {
        // do something.
    }
}
```

كود VB.NET

```
Public Class BMW
    Implements ITruck
    Implements ICar
    Private Sub move() Implements ICar.move
        ' do something.
    End Sub
    Private Sub move() Implements ITruck.move
        ' do something.
    End Sub
End Class
```

وبعد عمل object من الفئة BMW، يمكن تحديد اي واحدة من الدالتين يتم استدعاءها بالشكل التالي:

كود C#

```
BMW ahmedCar=new BMW();
ICar temp= (ICar)ahmedCar;
temp.move();
```

كود VB.NET

```
Dim ahmedCar As New BMW()
Dim temp As ICar = DirectCast(ahmedCar, ICar)
temp.move()
```

3. IEnumerable Interface

نواصل الآن دروسنا مع ال interfaces ، سنتعرض الآن ولثلاث دروس قادمة عن كيفية عمل فئة أو Class يقوم بعمل implement ل interface معين، في هذا الدرس سنبدأ ب interface المسمى IEnumerable وما هي الفائدة المرجوة وراء ذلك.

ملاحظة

أرجو منك أن تركز جيداً في هذا الدرس ، فهذا واحد من الدروس التي ستلاحقنا كثيراً أثناء تطبيق مبادئ LINQ

لنفترض ان لدينا فئة مصفوفة أسماء تحتوي على عدة بيانات بالشكل التالي

كود C#

```
string[] users = { "Ahmed", "Mohammed" };
```

كود VB.NET

```
Dim users As String() = {"Ahmed", "Mohammed"}
```

والآن لقراءة محتويات المصفوفة كنا نستخدم الكود التالي باستخدام for each كما تعلمنا سابقاً:

كود C#

```
foreach (string us in users)
{
    Console.WriteLine(us);
}
```

كود VB.NET

```
For Each us As String In users
    Console.WriteLine(us)
Next
```

هذا لو كنا نعتمد على array، تجد اننا نستخدم دالة للدوران على كافة العناصر ، سنقوم بتوسيع المثال نوعاً ليكون ذلك بالاعتماد على فئة ، سنفترض فئة الموظفين بالشكل التالي:

C#	كود

```
public class employee
{
    int emp_number;
    string emp_name;
}
```

VB.NET	كود

```
Public Class employee
    Private emp_number As Integer
    Private emp_name As String
End Class
```

وليكن لدينا فئة أخرى يحتوي على بيانات المستخدمين ، حيث ان المستخدم هو عبارة عن مجموعة من الموظفين بالشكل التالي مثلاً:

C#	كود

```
public class users
{
    employee[] emp = new employee[3];
    users()
    {
        emp[0] = new employee(10, "Ahmed");
        emp[1] = new employee(20, "Khaled");
        emp[2] = new employee(30, "Ali");
    }
}
```

VB.NET	كود

```
Public Class users
    Private emp As employee() = New employee(2) {}
    Private Sub New()
        emp(0) = New employee(10, "Ahmed")
        emp(1) = New employee(20, "Khaled")

        emp(2) = New employee(30, "Ali")
    End Sub
End Class
```

يمكننا استعراض السيارات ايضاً باستخدام for each ايضاً باستخدام جملة كالتالي:

C#	كود

```
foreach (employee e in emp)
{

}
```

كود | VB.NET

```
For Each e As employee In emp

Next
```

مع كل loop سيتم طباعة عنصر معين، هذا هو المبدأ الذي سنسعى إليه مع الـIEnumerable.

الشكل الاساسي لل interface المسمى IEnumerable:

كود | C#

```
public interface IEnumerable
{
    IEnumerator GetEnumerator();
}
```

كود | VB.NET

```
Public Interface IEnumerable
    Function GetEnumerator() As IEnumerator
End Interface
```

وكما ترى في المثال السابق ، يقوم بتعريف واجهة interface من نوع IEnumerator بالشكل التالي

كود | C#

```
public interface IEnumerator
{
    bool MoveNext();
    object Current { get; }
    void Reset();
}
```

كود | VB.NET

```
Public Interface IEnumerator
    Function MoveNext() As Boolean
    ReadOnly Property Current() As Object
    Sub Reset()
End Interface
```

لذا لو كنا نرغب في ان نجعل فئة users تقوم بعمل implements لهذه الواجهة interface سنضطر لعمل implement للدالة GetEnumerator() ، سيكون ذلك بالشكل التالي:

كود C#

```
public class users : IEnumerable
{
    employee[] emp = new employee[4];
    users()
    {
        emp[0] = new employee(10, "Ahmed");
        emp[1] = new employee(20, "Khaled");
        emp[2] = new employee(30, "Ali");
        emp[3] = new employee(40, "Sami");
    }
    public IEnumerator GetEnumerator()
    {
        return emp.GetEnumerator();
    }
}
```

كود VB.NET

```
Public Class users
    Implements IEnumerable
    Private emp As employee() = New employee(3) {}
    Private Sub New()
        emp(0) = New employee(10, "Ahmed")
        emp(1) = New employee(20, "Khaled")
        emp(2) = New employee(30, "Ali")
        emp(3) = New employee(40, "Sami")
    End Sub

    Public Function GetEnumerator() As IEnumerator
        Return emp.GetEnumerator()
    End Function
End Class
```

يمكننا القيام بذلك يدوياً ايضاً بالشكل التالي:

كود C#

```
IEnumerator i = emp.GetEnumerator();
i.MoveNext();
employee myCar = (employee)i.Current;
```

كود VB.NET

```
Dim i As IEnumerator = emp.GetEnumerator()
        i.MoveNext()
Dim myCar As employee = DirectCast(i.Current, employee)
```

ماذا نستفيد من هذه الطريقة ؟

في الواقع فأي مبرمج لقواعد البيانات يستطيع ان يستنتج العلاقة بسهولة ، إن ما نحن بصدده هنا هو مثال عن فئة تحتوي على مجموعة من البيانات اشبه بقاعدة بيانات ، واصبح باستطاعنا الآن القراءة منها والتنقل فيها باستخدام moveNext .

- استخدام yield

بدلاً من الكود السابق قم بكتابة الكود التالي:

كود C#
```
public IEnumerator GetEnumerator()
{
    yield return emp[0];
    yield return emp[1];
    yield return emp[2];
    yield break;
}
```

في هذه الحالة ومع اول استدعاء للدالة سيتم اعادة القيمة الأولى ، ثم الثانية فالثالثة وهكذا حتى نصل إلى yield break.

نستعرف على واحدة من أهم الاستخدامات الفعلية عندما نصل إلى LINQ ، وحتى ذلك الحين أتمنى ألا تغفل عينك عن هذه الواجهة .

4. ICloneable Interface

عملية ال clone يقصد بها عملية النسخ للكائن ، بمعنى اننا عندما نكتب الكود التالي:

كود C#
```
int x = 5;
int y = x;
```

كود VB.NET
```
Dim x As Integer = 5
Dim y As Integer = x
```

فإننا فعلياً نقوم بنقل قيمة X إلى y ، ولكن في حالة تعاملنا مع الفئات بالشكل التالي:

كود C#
```
Car x = new Car("BMW");
Car y = x;
```

كود VB.NET
```
Dim x As New Car("BMW")
Dim y As Car = x
```

في الواقع ان قيمة X لا تنتقل إلى y ، بل إن y يصبح يشير إلى المكان الذي فيه قيمة X ، ولذا فإن اي تعديل في قيمة X سيعدل قيمة y والعكس بالعكس.

من هنا كانت عملية ال clone من اجل عمل نسخة جديدة في القيمة ووضعها في المتغير الآخر.

الواجهة ICloneable التي تحتوي على الدالة clone، يمكن استخدامها لهذا الغرض، حيث يمكننا كتابة كود كالتالي:

كود C#
```
public object Clone()
{
    return new Car(this.carName);
}
```

كود VB.NET
```
Public Function Clone() As Object
    Return New Car(Me.carName)
End Function
```

الآن اصبح بامكانك عمل نسخة جديدة باستخدام Clone بالشكل التالي:

كود C#
```
Car x = new Car("BMW");
Car y = (Car)x.Clone();
```

كود VB.NET
```
Dim x As New Car("BMW")
Dim y As car = DirectCast(x.Clone(), Car)
```

5. Interface IComparable

كما هو واضح من الاسم ، يستخدم هذا ال interface للمقارنة ، ولذا من الطبيعي ان يكون شكل هذا ال interface بالشكل التالي:

كود C#

```
public interface IComparable
{
    int CompareTo(object o);
}
```

كود VB.NET

```
Public Interface IComparable
    Function CompareTo(ByVal o As Object) As Integer
End Interface
```

يمكنك ملاحظة ان الدالة تعيد قيمة رقمية، فعلياً هي تعيد في المعتاد القيم 0 في حالة التساوي ، القيمة 1 في حالة كون الطرف الأول اكبر ، و -1 في حالة كون الطرف الأول أصغر.

طبعاً لا داعي من اعادة تذكيرك بأن الكائنين يكونان متساويان ليس لاعتبارات تساوي قيمهم الداخلية بل النقطة في اشارتهم لنفس المكان في الذاكرة ، وهو ما لن نحتاجه ، حيث اننا نحتاج لمقارنة فعلية . وهذا ما قمنا به من خلال الدالة Equals.

اما النقطة التي لدينا هنا فلسنا بحاجة لمعرفة هل يساوي هذا الكائن ذاك ام لا ، بل نحن في حاجة إلى معرفة ايهما اكبر او اصغر ، ولنفترض تاريخ صنع السيارة ضمن الفئة الخاصة بها.

ببساطة كما قمنا في الدرس السابق ، سنجعل الفئة الخاصة بنا تطبق الفئة IComparable، ثم نكتب بعض الاكواد في الدالة CompareTo، لنفترض الشكل التالي مثلاً:

كود C#

```
int IComparable.CompareTo(object obj)
{
    Car temp = (Car)obj;
    if (this.year > temp.year)
        return 1;
    if (this.year < temp.year)
        return -1;
    else
        return 0;
}
```

كود	VB.NET

```
Private Function CompareTo(ByVal obj As Object) As Integer Implements
IComparable.CompareTo
    Dim temp As Car = DirectCast(obj, Car)
    If Me.year > temp.year Then
        Return 1
    End If
    If Me.year < temp.year Then
        Return -1
    Else
        Return 0
    End If
End Function
```

طبعاً لا داعي لتذكيرك بانك تستطيع اعادة -100 و 0 و 94 بدلاً من -1 و 0 و 1 ، لكن هذا لتسهيل المفهوم ، الدالة تعمل بدلالة رقم سالب يعني اصغر ، صفر يعني التساوي ورقم موجب يعني اكبر فقط دون ان يعنيها قيمة هذا الرقم الفعلية.

الدالة Sort

لنفترض ان لدينا مصفوفة array من الارقام باسم myNumbers، في هذه الحالة لو قمنا بكتابة الأمر التالي لكان الكود مفهوماً:

كود	C#

```
Array.Sort(myNumbers);
```

كود	VB.NET

```
Array.Sort(myNumbers)
```

ولكن لو كان لدينا مصفوفة من السيارات ، وكتبنا هذا الكود لترتيبها فإن هذا بالتأكيد سيحدث خطأ.

لكن في حالة كون الفئة car تطبق ال interface IComparable، ففي هذه الحالة سيمكنك استخدام هذا الدالة ببساطة لإنها سوف تقوم بالمقارنة اعتماداً على دالة CompareTo التي قمت انت ببرمجتها ، وربما يكون هذا سبباً كافياً لتقوم بتطبيق هذا ال interface وكتابة كود الدالة بدلاً من كتابة دالة باسم check مثلاً او اي اسم آخر للقيام بنفس المهمة . هذا طبعاً بالاضافة لتسهيل استخدام الفئة الخاصة بكل لاحقاً ووضوحها لأي مبرمج آخر.

الباب

10

الأخطاء واقتناصها

سنتعرف في درسنا هذا على الأخطاء وكيفية اقتناصها وتفاديها ، معاني رسائل الخطأ واشهر الاخطاء وخلافه ، ولكن قبل البداية نحتاج لإن نوضح أقسام الأخطاء التي تحصل في اي برنامج:

1. الأخطاء النحوية Syntax Errors

هذا النوع من الاخطاء هو الاسهل ، وفي Advanced Programming Environments مثل الVisual Studio , NetBeans ... etc، يتم اكتشاف هذه الأخطاء فورياً ، مثال هذا الخطأ كتابة الجملة التالية:

كود	VB.NET

```
if x.Nome = somevalue
```

بالطبع ستجد رسالة خطأ قبل التنفيذ تخبرك بأن الخاصية Nome غير موجودة، مثل هذه الأخطاء هي الاسهل ويتم اكتشافها من خلال بيئة لغة البرمجة التي تعمل عليها، وفي Visual Studio .net 2008 اصبحت رسائل الخطا واضحة للغاية ويمكن تفسيرها بسهولة وحلها بهذه الطريقة .

2. الأخطاء المنطقية Logical Errors

هذا النوع من الأخطاء هو الأصعب، فعلى صعيد كتابة الكود ربما لا يوجد خطأ نحوي ولكنه خطا منطقي يظهر عند التنفيذ، ابسط مثال على هذا الخطأ هو كتابة كود كالتالي:

كود	VB.NET

```
Dim x as Byte= 100000
```

طبعاً تعرف ان حدود النوع Byte اصغر من هذا الحد، ولكن في الاصدارات القديمة لم يكن هذا ليظهر خطأ حيث ان الجملة مكتوبة نحوياً كما ترى .

امثلة على هذا الخطأ اسناد قيمة ل object قبل عمل new له .. الخ.

لكن مع اصدارات فيجوال ستوديو الجديدة ، اصبحت مثل هذه الأخطاء تظهر مباشرة ، بل ان هناك انواعاً اصعب من ال Logic Errors اصبح الفيجوال ستوديو قادر على اكتشافها على شكل warnings.

لن نذهب بعيداً ، سنبدأ بتقسيم ال Logical Errors وهي الأخطاء الأهم إلى ثلاثة أنواع اساسية:

- User Error

أخطاء تنتج من استخدام البرنامج ، لو افترضنا المثال السابق ل Byte نقوم فيه بتخزين عمر المستخدم ، لكن المستخدم قام بادخال رقم 10 الاف ، هذا الخطأ من المستخدم سيتسبب في المشاكل لك فيما لو لم تكن قد اضفت شرط التأكد من عدم تجاوز العمر لحد معين، ايضاً ادخال بيانات نصية في خانة العمر وخلافه تندرج تحت اسم أخطاء المستخدم.

- Exceptions

النوع الأشهر من الأخطاء ، محاولة فتح ملف او قاعدة بيانات غير موجودة مثلاً حيث لم يتم تحميلها بصورة صحيحة ، محاولة قراءة بيانات من قاعدة البيانات في حين انها تساوي null بدون استخدام nullable type ، محاولة الكتابة إلى ملف نصي ReadOnly، وخلافه من الأخطاء المشهورة.

- Bugs

اكثر الأخطاء شهرة ، لا يمكن حصرها ولا عدها ، وتوجد في جميع البرامج بما فيهم نسخة الويندوز التي تستخدمها ، في العادة لن يخلو برنامج منها ولكننا نحاول تفاديها قدر المستطاع، قد تحدث بسبب نسيان حذف متغير او قراءة متغير من قيمة موجودة اصلاً في الذاكرة ونحن نظن انها قيمة فارغة ... الخ ، هذه الأخطاء قد لا تظهر ل 99% من المستخدمين ولكنا تظهر لمستخدم واحد فقط ، لذا في العادة تكون هناك عدة نسخ تجريبية من اي برنامج لمحاولة معرفة اماكن امثال هذه الأخطاء وتعديلها قبل طرح النسخة الرسمية.

في درسنا هذا سنركز على النوع الثاني من الأخطاء وهو الأهم ، النوع الأول ايضاً سنحاول وضع استنثناءات من اجل التأكد من اختيارات المستخدم ولكن جمل التحقق هي الأهم في الحالة الأولى، اما الحالة الثالثة فالتجربة المستمرة والمتابعة هي الوسيلة الأمثل لتقليلها.

3. الفئة System.Exception

هي الفئة المختصة في عالم net. بالتعامل مع الأخطاء التي تحدث في النظام ، في الواقع فإن أي خطأ يرسل للنظام ثم يقوم النظام بارساله إلى ال CLR، والذي بدوره يخول System.Exception للتعامل مع هذا الخطا ، محتويات هذه الفئة بالشكل التالي:

كود C#

```
public class Exception : ISerializable, _Exception
{
    // Public constructors
    public Exception(string message, Exception innerException);
    public Exception(string message);
    public Exception();
    // Methods
    public virtual Exception GetBaseException();
    public virtual void GetObjectData(SerializationInfo info,
    StreamingContext context);
    // Properties
    public virtual IDictionary Data { get; }
    public virtual string HelpLink { get; set; }
    public System.Exception InnerException { get; }
    public virtual string Message { get; }
    public virtual string Source { get; set; }
    public virtual string StackTrace { get; }
    public MethodBase TargetSite { get; }
}
```

كود	VB.NET

```
Public Class Exception
    Implements ISerializable
    Inherits _Exception
    ' Public constructors
    Public Sub New(ByVal message As String, ByVal innerException As Exception)
    End Sub
    Public Sub New(ByVal message As String)
    End Sub
    Public Sub New()
    End Sub
    ' Methods
    Public Overridable Function GetBaseException() As Exception
    End Function
    Public Overridable Sub GetObjectData(ByVal info As SerializationInfo, ByVal
context As StreamingContext)
    End Sub
    ' Properties
    Public Overridable ReadOnly Property Data() As IDictionary
        Get
        End Get
    End Property
    Public Overridable Property HelpLink() As String
        Get
        End Get
        Set(ByVal value As String)
        End Set
    End Property
    Public ReadOnly Property InnerException() As System.Exception
        Get
        End Get
    End Property
    Public Overridable ReadOnly Property Message() As String
        Get
        End Get
    End Property
    Public Overridable Property Source() As String
        Get
        End Get
        Set(ByVal value As String)
        End Set
    End Property
    Public Overridable ReadOnly Property StackTrace() As String
        Get
        End Get
    End Property
    Public ReadOnly Property TargetSite() As MethodBase
        Get
        End Get
    End Property
End Class
```

سنحاول التعرف على الخصائص والطرق الاساسية لهذه الفئة :

الخاصية	المعنى
Message	رسالة الخطأ الحاصلة
Source	ملف الاسمبلي الذي قام بعمل throw لهذا الخطأ
HelpLink	حتوي هذه الخاصية على رابط يشرح المشكلة ببعض التفصيل ، تستطيع الاستفادة منه كمبرمج وربما يستفيد منه المستخدم المتخصص في ال IT لنظامك

الجدول 10. 1. بعض خصائص الفئة Exception

4. رمي الاستثناءات Throwing Exceptions

خلال تنفيذك للبرنامج يمكن ان يقوم البرنامج بعمل throw لخطأ ما مثل عدم وجود ملف معين. سنتعلم كيفية قراءة هذا الخطأ والتعامل معه ، لكن تظل هناك حالة أخرى ترغب انت فيها بعمل throw للخطأ ، لنفترض انك تقوم بعمل check تتأكد من عدم وجود الملف ومن ثم تقوم بعرض رسالة خطأ في حالة عدم وجود الملف بالشكل التالي مثلاً:

كود C#

```
if (!System.IO.File.Exists("c:\\ahmed.txt"))
{
    Console.WriteLine("there is no file");
}
```

كود VB.NET

```
If Not System.IO.File.Exists("c:\ahmed.txt") Then
    Console.WriteLine("there is no file")
End If
```

لكنك ربما لم تقم باقتناص خطأ كون الملف للقراءة فقط ، وترغب في صورة قنص اخطاء موحدة ، أو لأي اعتبارات أخرى ترغب في عمل throw للمبرمج مثلاً لو كنت تبرمج فئة ليتم استخدامها . لأي من هذه الاعتبارات يمكن عمل throw لخطأ بالشكل التالي:

كود	C#

```
if (!System.IO.File.Exists("c:\\ahmed.txt"))
{
    throw new Exception("there is no file");
}
```

كود	VB.NET

```
If Not System.IO.File.Exists("c:\ahmed.txt") Then
    Throw New Exception("there is no file")
End If
```

هكذا تجد ان الفيجوال ستوديو قام باظهار خطأ بالرسالة التي اوضحتها ، تستطيع لاحقاً قراءة كائن الخطأ والتعامل معه كما تتعامل مع الخطأ الذي ينتج تلقائياً من الفيجوال بيسك.

5. اقتناص الأخطاء Catching Exceptions

الآن جاء دور اقتناص الأخطاء ، ابسط طريقة هي باستخدام Try بحيث يخرج البرنامج من البلوك في حالة وجود الخطأ دون ان يتسبب في توقف البرنامج ، بالشكل التالي مثلاً:

كود	C#

```
try
{
    x += 100;
    console.writeLine("no errror");
}
catch
{
    console.writeLine("some error!");
}
```

كود	VB.NET

```
Try
    x += 100
    Console.WriteLine("no errror")
Catch
    Console.WriteLine("some error!")
End Try
```

في حالة وجود خطأ في عملية الجمع السابقة فسيتم مباشرة الانتقال إلى catch ، فيما عدا ذلك سيواصل البرنامج دون المرور عليها ، وفي كل الاحيان لن يتم ايقاف البرنامج.

يمكننا ايضاً عرض تفاصيل عن الخطأ الذي حدث باستخدام الخصائص السابقة

C# كود

```
try
{
    x += 100;
    console.writeLine("no errror");
}
catch (Exception e)
{
    Console.WriteLine("Method: {0}", e.TargetSite);
    Console.WriteLine("Message: {0}", e.Message);
    Console.WriteLine("Source: {0}", e.Source);
}
```

VB.NET كود

```
Try
    x += 100
    Console.WriteLine("no errror")
Catch e As Exception
    Console.WriteLine("Method: {0}", e.TargetSite)
    Console.WriteLine("Message: {0}", e.Message)
    Console.WriteLine("Source: {0}", e.Source)
End Try
```

لو كنا نعرف بعض الأخطاء التي يمكن ان تحدث، فيمكننا اختبارها وعرض الرسالة فيما عدا ذلك، لنفترض المثال التالي في حالة كوننا نعرف أن الخطأ يمكن ان يكون بسبب تجاوز المجال overflow، وفيما عدا ذلك سنظهر رسالة بخطأ عام:

C# كود

```
try
{
    x += 100;
    console.writeLine("no errror");
}
catch (OverflowException e0)
{
    Console.WriteLine("value of x more than up bound");
}
catch (Exception e)
{
    Console.WriteLine("Method: {0}", e.TargetSite);
    Console.WriteLine("Message: {0}", e.Message);
    Console.WriteLine("Source: {0}", e.Source);
}
```

كود VB.NET

```
Try
    x += 100
    Console.WriteLine("no errror")
Catch e0 As OverflowException
    Console.WriteLine("value of x more than up bound")
Catch e As Exception
    Console.WriteLine("Method: {0}", e.TargetSite)
    Console.WriteLine("Message: {0}", e.Message)
    Console.WriteLine("Source: {0}", e.Source)
End Try
```

5. 1. استخدام Finally

تستخدم للتنفيذ بعد نهاية البلوك try - catch ويتم تنفيذها في حالة وجود خطأ او عدمه، فمثلاً لو كنا نرغب في طباعة نص ما بغض النظر عن حدوث خطأ في المتغير X من عدمه نكتب الكود التالي:

كود C#

```
try
{
    x += 100;
    console.writeLine("no errror");
}
catch (Exception e)
{
    Console.WriteLine("Method: {0}", e.TargetSite);
    Console.WriteLine("Message: {0}", e.Message);
    Console.WriteLine("Source: {0}", e.Source);
}
finally
{
    Console.WriteLine("somehing");
}
```

كود VB.NET

```
Try
    x += 100
    Console.WriteLine("no errror")
Catch e As Exception
    Console.WriteLine("Method: {0}", e.TargetSite)
    Console.WriteLine("Message: {0}", e.Message)
    Console.WriteLine("Source: {0}", e.Source)
Finally
    Console.WriteLine("somehing")
End Try
```

5. 2. استخدام break

للخروج من الاستثناء في مرحلة ما ، يمكن استخدام break أو Exit Try بالنسبة للفيجوال بيسك.

5. 3. استخدام Target Site

توفر هذه الخاصية معلومات عديدة حول الفئة والدالة التي قامت بعمل throw للخطأ، يمكن الاستفادة منها في عمل Debug لمعرفة مكان حدوث الخطأ.

5. 4. استخدام HelpLink

تستطيع افادة مستخدم الفئة أو المستخدم بها ، عن طريق وضع لينك معين يمكنه الاستفادة منه بالشكل التالي مثلاً:

كود | C#

```
try
{
    m += 100;
    Console.WriteLine("no errror");
}
catch (Exception e)
{
    e.HelpLink = "www.ahmedgamal-space.blogspot.com";
}
```

كود | VB.NET

```
Try
    m += 100

    Console.WriteLine("no errror")
Catch e As Exception

    e.HelpLink = "www.ahmedgamal-space.blogspot.com"
End Try
```

6. عمل أخطاء خاصة

يمكنك تعريف فئة من النوع خطأ يمكنك استخدامها لاحقاً في برامجك يتم اشتقاقها من الفئة ApplicationException كما أنك تحتاج أيضاً لجعلها مشتقة من الفئة Serializable من اجل امكانية استخدامها في ال Remoting وخلافه ، يمكنك تعريف خطأ خاص بالشكل التالي مثلاً :

كود C#

```
public class NewException : ApplicationException, ISerializable
{
    public NewException()
    {
        // something here
    }
    public NewException(string message)
    {
        // something here
    }
    public NewException(string message, Exception inner)
    {
        // something here.
    }
    // ال لعملية Serilization
    protected NewException(SerializationInfo info, StreamingContext context)
    {
        // Add something here.
    }
}
```

كود	VB.NET

```
Public Class NewException
    Inherits ApplicationException
    Implements ISerializable
    Public Sub New()
        ' Add something here.
    End Sub
    Public Sub New(ByVal message As String)
        ' Add something here.
    End Sub
    Public Sub New(ByVal message As String, ByVal inner As Exception)
        ' Add something here.
    End Sub
    ' ال لعملية Sterilization
    Protected Sub New(ByVal info As SerializationInfo, ByVal context As
StreamingContext)
        ' Add something here.
    End Sub
End Class
```

للمزيد حول الاخطاء واقتناصها يمكنك الرجوع إلى MSDN:

رابط

http://msdn.microsoft.com/en-us/library/ms229014(VS.80).aspx

الباب

11

التجميعات Collections

في تطبيقاتنا البرمجية المختلفة عادة ما نلجأ لعمل container يحتوي على عدة عناصر، من أجل ذلك كانت ال array ورأينا ان بامكاننا عمل array من الارقام والنصوص واخيراً حتى من فئات مختلفة. إلا اننا قد نحتاج إلى بعض المهام الاكثر من تلك التي توفرها لنا ال array، من اجل هذا وجدت ال collections المختلفة، في درسنا هذا سوف نتعرف عليها إن شاء الله.

1. الInterfaces في System.Collections

لو راجعنا الواجهات الموجودة في ال collection لوجدنا الواجهات التالية:

ICollection:

تحتوي هذه الواجهة على بعض النقاط الاساسية مثل ال Size وال IsSynchronized وخلافه ، لها الشكل العام التالي:

كود C#
```
public interface ICollection : IEnumerable
{
    int Count { get; }
    bool IsSynchronized { get; }
    object SyncRoot { get; }
    void CopyTo(Array array, int index);
}
```

كود VB.NET
```
Public Interface ICollection
    Inherits IEnumerable
    ReadOnly Property Count() As Integer
    ReadOnly Property IsSynchronized() As Boolean
    ReadOnly Property SyncRoot() As Object
    Sub CopyTo(ByVal array As Array, ByVal index As Integer)
End Interface
```

IComparer: تعرفنا عليها سابقا.

IDictionary:

تستخدم كما هو واضح من اسمها من اجل الادلة، يمكن استخدامها لتخزين مثلاً بيانات الموظفين وعناوينهم بحيث يمكن العثور بعد ذلك على الشخص بدلالة اسمه او عنوانه ، تحتوي على دوال للاضافة والحذف وخلافه . الشكل العام لها بالشكل التالي :

كود	C#

```
public interface IDictionary :ICollection, IEnumerable
{
    bool IsFixedSize { get; }
    bool IsReadOnly { get; }
    object this[object key] { get; set; }
    ICollection Keys { get; }
    ICollection Values { get; }
    void Add(object key, object value);
    void Clear();
    bool Contains(object key);
    IDictionaryEnumerator GetEnumerator();
    void Remove(object key);
}
```

كود	VB.NET

```
Public Interface IDictionary
    Inherits ICollection
    Inherits IEnumerable
    ReadOnly Property IsFixedSize() As Boolean
    ReadOnly Property IsReadOnly() As Boolean
    Default Property Item(ByVal key As Object) As Object
    ReadOnly Property Keys() As ICollection
    ReadOnly Property Values() As ICollection
    Sub Add(ByVal key As Object, ByVal value As Object)
    Sub Clear()
    Function Contains(ByVal key As Object) As Boolean
    Function GetEnumerator() As IDictionaryEnumerator
    Sub Remove(ByVal key As Object)
End Interface
```

IEnumerable:

تم شرحها سابقاً.

IEnumerator:

تم شرحها سابقاً.

IDictionaryEnumerator:

لو لاحظت في IDictionary ستجد الدالة GetEnumerator ، هذا هو ال interface الخاص بهذه الدالة ، لها الشكل العام التالي:

كود C#

```
public interface IDictionaryEnumerator : IEnumerator
{
    DictionaryEntry Entry { get; }
    object Key { get; }
    object Value { get; }
}
```

كود VB.NET

```
Public Interface IDictionaryEnumerator
    Inherits IEnumerator
    ReadOnly Property Entry() As DictionaryEntry
    ReadOnly Property Key() As Object
    ReadOnly Property Value() As Object
End Interface
```

:IHashCodeProvider

يختص باعادة ال hash code لل collection المعين باستخدام الدالة GetHashCode.

:IList

قائمة ، هذا كل ما في الأمر ، تحتوي على عدة دوال تسهل التعامل معها للاضافة والبحث والحذف وخلافه الشكل العام لها بالشكل التالي:

كود C#

```
public interface IList :ICollection, IEnumerable
{
    bool IsFixedSize { get; }
    bool IsReadOnly { get; }
    object this[int index] { get; set; }
    int Add(object value);
    void Clear();
    bool Contains(object value);
    int IndexOf(object value);
    void Insert(int index, object value);
    void Remove(object value);
    void RemoveAt(int index);
}
```

VB.NET | كود

```
Public Interface IList
    Inherits ICollection
    Inherits IEnumerable
    ReadOnly Property IsFixedSize() As Boolean
    ReadOnly Property IsReadOnly() As Boolean
    Default Property Item(ByVal index As Integer) As Object
    Function Add(ByVal value As Object) As Integer
    Sub Clear()
    Function Contains(ByVal value As Object) As Boolean
    Function IndexOf(ByVal value As Object) As Integer
    Sub Insert(ByVal index As Integer, ByVal value As Object)
    Sub Remove(ByVal value As Object)
    Sub RemoveAt(ByVal index As Integer)
End Interface
```

2. الفئات في System.Collections

يحتوي على الفئات التالية:

- ArrayList
- Hashtable
- Queue
- SortedList
- Stack

2. 1. ArrayList

هي قائمة array كما هو واضح من الاسم ، تتميز بنفس سمات ال array من حيث تخزينها لعدة بيانات، اضافة لمميزات القائمة التي تسهل عمليات الاضافة والتعديل والترتيب والحذف وخلافه داخل عناصر هذه المصفوفة.

لنفترض ان لدينا فئة (سيارة) السابقة والتي تحتوي على اسم السيارة وموديلها ، كنا نعرف مصفوفة منها باستخدام الأمر التالي:

C# | كود

```
Car[] carArray = new Car[4];
```

VB.NET | كود

```
Dim carArray As Car() = New Car(3)
```

ربما لم تكن تعرف انك تحتاج لاربع سيارات فقط ، ربما تحتاج للزيادة او للنقصان ، ربما ترغب بحذف واحدة منهم من المنتصف ... الخ ، فكما ترى لا توفر لنا ال array الكثير من الخيارات لتسهيل التعامل مع هذه الخصائص ، ستحتاج لبعض الأوامر للحذف وخلافه.

بداية قم بعمل using ل System.Collection لتكون قادراً على التعامل المباشر معها لاحقاً.

كود — C#

```
using System.Collections;
```

كود — VB.NET

```
Imports System.Collections
```

لكن جاءت ArrayList لتلقي عن كاهلك اي مشاكل بخصوص هذه العمليات ، كل ما عليك هو تعريف ArrayList بالطريقة التالية:

كود — C#

```
ArrayList carList = new ArrayList();
```

كود — VB.NET

```
Dim carList As New ArrayList()
```

والآن يمكنك اضافة سيارة:

كود — C#

```
Car temp = new Car("BMW", 1990);
carList.Add(temp);
```

كود — VB.NET

```
Dim temp As New Car("BMW", 1990)
carList.Add(temp)
```

تستطيع ايضاً عمل Insert في جزء معين، لاضافة السيارة مثلاً في المكان الثالث نكتب الكود التالي:

كود — C#

```
carList.Insert(temp);
```

كود | VB.NET

```
carList.Insert(temp)
```

في الناحية الأخرى ، تستطيع مسح كامل القائمة:

كود | C#

```
carList.Clear();
```

كود | VB.NET

```
carList.Clear()
```

أو حذف عنصر معين بدلالة موقعه:

كود | C#

```
carList.RemoveAt(4);
```

كود | VB.NET

```
carList.RemoveAt(4)
```

او بدلالة العنصر نفسه :

كود | C#

```
carList.Remove(temp);
```

كود | VB.NET

```
carList.Remove(temp)
```

يمكننا استخدام الخاصية Count لمعرفة العدد، وبالتالي الدوران عليهم جميعاً بالشكل التالي مثلاً:

كود | C#

```
for (int i = 0; i < carList.Count; i++)
{
    temp = (Car)carList[i];
}
```

كود	VB.NET

```
For i As Integer = 0 To carList.Count - 1
    temp = DirectCast(carList(i), Car)
Next
```

يمكننا استخدام بعض الخصائص الأخرى مثل Sort كما تعلمنا سابقاً، يمكننا تحويلها إلى array عادية باستخدام الدالة ToArray، الدالة Reverse ستعكس ترتيب القائمة، الدالة IndexOf تستخدم للبحث بالطريقة التالية، لنفترض لدينا السيارة temp ونرغب في البحث عنها ضمن السيارات الموجودة ، سيكون ذلك بالشكل التالي:

كود	C#

```
int find = carList.IndexOf(temp, 0);
```

كود	VB.NET

```
Dim find As Integer = carList.IndexOf(temp, 0)
```

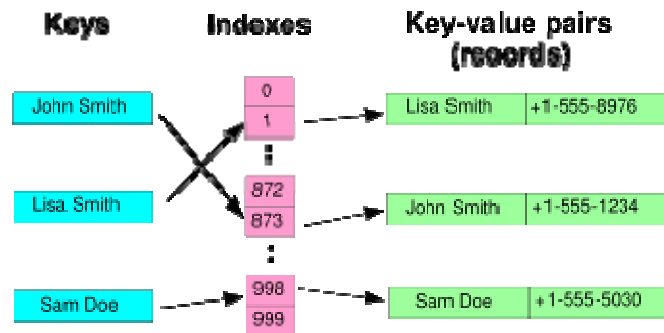
وهكذا تستطيع استخدام المتغير Find للوصول إلى مكان العنصر والتعامل معه كما تريد ، كما يمكنك تحديد نقطة بداية البحث 0 كما حددنا او تغييرها او عدم استخدامها اصلاً .

2. 2. HashTable

واحد من طرق تخزين البيانات Data Structure، مهمته الاساسية تسهيل البحث عن البيانات المرتبطة ، إذا لم تكن قد مرت عليك هذه ال Hash Table من قبل فانصحك بالتعرف عليها من خلال ويكبيديا هنا:

رابط

http://en.wikipedia.org/wiki/Hash_table

وباختصار شديد ، فال HashTable هي عبارة عن مجموعة من المفاتيح keys يخزن معها من الدلائل Indexes تشير إلى مكان باقي القيم المرتبطة مع المتاح الاساسي.

الصورة 11. 1. بنية ال HashTable

في السابق كنا نقوم بتكوين ال HashTable عبارة عن Linked List يرتبط كل عنصر فيه بLinked List اخرى، وكنا نقوم بعمل دواله واجراءاته المختلفة، أما الآن فيمكنك ببساطة تعريف HashTable والبدء في وضع البيانات فيه مباشرة باستخدام الأوامر التقليدية Add,AddRange و Remove وخلافه مما تعلمناه في الدرس السابق.

2. 3. Queue

الترجمة الحرفية له في الطابور ، ومع اني لا استسيغ بعض الترجمات لكن لا بأس من توضيح ان ال Queue ما هو إلا طابور فعلاً ، حيث يتم دفع البيانات إليه باستخدام Enqueue واخراجها باستخدام Dequeue، أول الداخلين هو اول الخارجين فنحن هنا نتحدث عن طابور حقيقي.

إذا كنت تبرمج برنامج لتلقي طلبات العملاء، فمن الطبيعي ان تضعها في طابور ليتم معالجة الاسبق بالوصول اولاً، في السيرفرات ايضاً، وربما اشهر امثلة ال Queue هو البروسيسور (المعالج) حيث يتم ترتيب المهمات ليتم ادخالها إلى المعالج، الأول وصولاً للطابور هو الأول تنفيذاً بابسط صورة ، لكن لا تنس انه في نظم المعالجات الحديثة هناك العديد من العناصر التي تتحكم في دخول البيانات مثل الاولوية اضافة لنظام ال Slides حيث لا يتم تنفيذ ال Task مرة واحدة ... الخ.

لا نريد ان نبتعد كثيراً، سنفترض طابور العملاء بالشكل التالي:

كود C#

```
Queue empQueue = new Queue();
employee temp = new employee();
for (int i = 0; i < 10; i++)
{
    temp.userName = Console.ReadLine();
    empQueue.Enqueue(temp);
}
```

كود VB.NET

```
Dim empQueue As New Queue()
Dim temp As New employee()
For i As Integer = 0 To 9
    temp.userName = Console.ReadLine()
    empQueue.Enqueue(temp)
Next
```

لو قمنا الآن بعمل Loop لنعمل Dequeue، ستجد ان اول الموظفين دخولاً سيكون أولهم خروجاً.

طبعاً لا تنس أن بامكانك استخدام بعض الخصائص مثل Count وخلافه تلك التي شرحناها في الدرس السابق.

قبل النهاية اشير فقط إلى ان الدالة Dequeue تعيد العنصر الأول وتقوم بحذفه مباشرة من الطابور ، أما الدالة Peek فهي تعيد اول عنصر في الترتيب ولكنها تحتفظ به في الطابور كما كان.

مع تطبيق هذه الفئة، أضن اظن انك تذكر لو كنت درست Data Structure وكيفية تطبيق ال Queue باستخدام الـ linked list ، وال circular queue والكثير مما لن تزعج نفسك به منذ الآن فصاعداً مع هذه الفئة .

2. 4. Stack

مثل ال Queue فيما عدا انه يتبع تقنية LIFO - Last Input First Output ، حيث ان العنصر الآخير في الدخول هو الأول في الخروج ، يتم ادخال العناصر باستخدام Push ويتم اخراجها باستخدام Pop ، ال Peek تقوم بمهمتها كما هي في عرض العنصر الأول وهو في حالتنا الآخر دخولاً دون حذفه من ال Stack.

تعريف ال Stack يتم ببساطة بالشكل التالي:

كود	C#

```
Stack jobStack = new Stack();
```

كود	VB.NET

```
Dim jobStack As New Stack()
```

الاستخدامات

يسهل تصور استخدامات من اجل ال Queue، ذلك ان معظم تطبيقات حياتنا اليومية تعتمد على ذلك ، حيث الواصل أولاً يخدم اولاً ، لكن ال Stack بمبدئة المنافي للعدالة ربما سيكون غريباً بعض الشيء تطبيقه في حياتنا الحقيقية ، لذا فإن استخدام ال Stack يقتصر في العادة على تطبيقات الكومبيوتر.

ابسط استخدام لل Stack هو حل المعادلات في البروسيسور، إن المعادلة:

A + B

تدخل إلى البروسيسور بالشكل التالي:

AB+

يسمى هذا الاسلوب باسم postfix، وفي البروسيسور يتم ادخالهم في Stack لتنفيذهم حيث ان تنفيذهم لن يتم بترتيب الوصول ، مثال آخر في ترجمة ال Compilers ايضاً.

في المرفقات برنامج يوضح كيفية التحويل بين postfix و infix كان احد مشاريع الكلية في الفرقة الثالثة ل FUTEX Group، يوضح البرنامج كيفية التحويل وكيفية يقوم البروسيسور بوضعهم في Stack وتنفيذهم ، قم باختيار View لاستعراض سرعة عرض الخطوات ، يمكنك الحل باستخدام الارقام او الرموز ويمكنك الاختيار فيما بينهما من اعلى النافذة . هذا إذا كنت مهتماً ببعض التفاصيل.

رابط

http://vb4arab.com/vb/uploaded/3/21207604046.rar

أو يمكنك الإطلاع على ناتج عملية التحويل مباشرة من هذا الموقع :

رابط

http://scriptasylum.com/tutorials/infix_postfix/infix_postfix.html

مزيد من التفاصيل تجدها في موقع MSDN هنا:

رابط

http://msdn2.microsoft.com/en-us/library/aa289149(VS.71).aspx

كل ما سبق في هذا الدرس ينطبق تحت علم Data Structure ، يمكنك الانطلاق من هذه النقطة إذا كنت ترغب في معرفة الكثير عن هذا العالم الذي يتداخل معك كثيراً في عالمك ، سيفيدك ان تكون مطلعاً على اساسياته اضافة لعلم ال Algorithms.

ايضاً هناك ما يعرف باسم System.Collections.Specialized ويحتوي على فئات اخرى اكثر تخصصاً مشتقة من الفئات السابقة.

الباب

12

البرمجة المتقدمة في ال

.net

في هذا الفصل سوف نتعرف على بعض عناصر البرمجة في net. منها الجديد الذي ظهر لأول مرة مع net 2008. فقط ومنها عناصر موجودة قبلاً ولكنها متقدمة نوعاً ما ، لذا قمنا بتجميعها سوية في هذا الباب .

1. ال Generics

نواصل في هذا الدرس مع واحدة من التطورات الجديدة مع net Framework 2.0.، هي ال generics لتعطي بعداً جديداً لمفهوم ال overloading، لنفترض مثالاً ما للطباعة بالشكل التالي:

كود C#

```
public static void print(int x)
{
    Console.WriteLine("Print As Integer {0}", x);
}
public static void print(long x)
{
    Console.WriteLine("Print As Long {0}", x);
}
public static void print(string x)
{
    Console.WriteLine("Print As String {0}", x);
}
```

كود VB.NET

```
Public Shared Sub print(ByVal x As Integer)
    Console.WriteLine("Print As Integer {0}", x)
End Sub
Public Shared Sub print(ByVal x As Long)
    Console.WriteLine("Print As Long {0}", x)
End Sub
Public Shared Sub print(ByVal x As String)
    Console.WriteLine("Print As String {0}", x)
End Sub
```

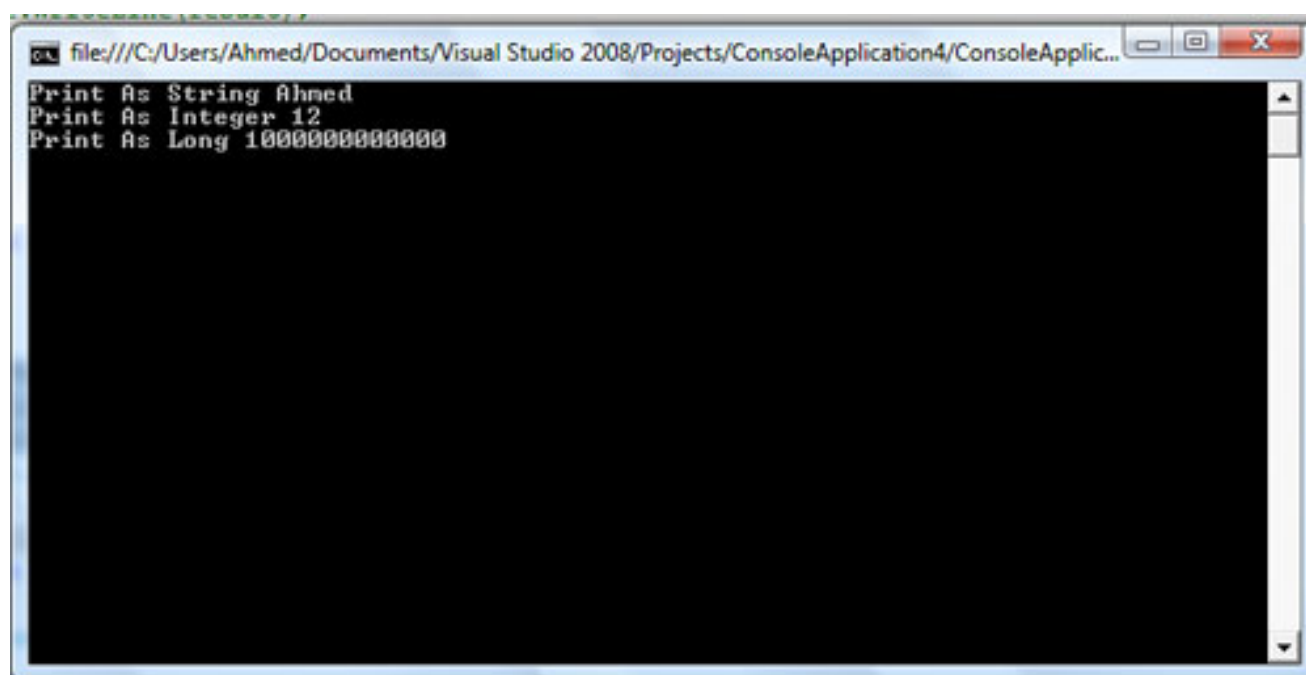
الكود كما هو واضح قام بعمل ثلاث دوال بنفس الاسم لاستقبال بارميترات مختلفة ، وهكذا تجد انك تستطيع في الكود استدعاء الدالة الطباعة للارقام او لل longs او للنصوص ، جرب عدة عمليات لطباعة انواع مختلفة وشاهد جملة الطباعة ، على سبيل المثال الأوامر التالية:

كود	C#

```
print("Ahmed");
print(12);
print(1000000000000);
```

كود	VB.NET

```
Print("Ahmed")
Print(12)
Print(1000000000000)
```

سيكون ناتج الطباعة بالشكل التالي:

الصورة 12. 1. ناتج تنفيذ العمليات

ال Generic ستغير المفهوم نوعاً ، حيث ان بامكانك الآن تعريف دالة لا تستقبل نوعاً معيناً من المتغيرات ، بل هي تستقبل <T> وتتعامل معه على هذا الاساس ، لذا قد نستطيع افتراض الدوال الثلاث السابقة بالشكل التالي

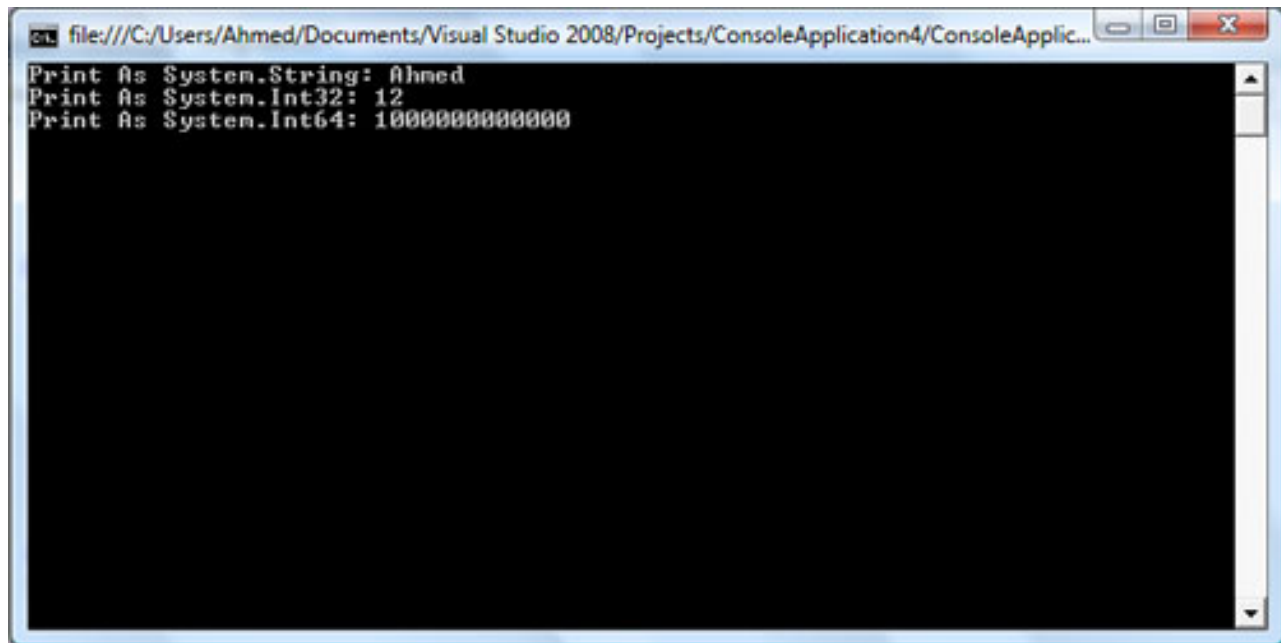
كود	C#

```
public static void print<T>(T x)
{
    Console.WriteLine("Print As {0}: {1}", typeof(T), x);
}
```

كود	VB.NET

```
Public Shared Sub print(Of T)(ByVal x As T)
    Console.WriteLine("Print As {0}: {1}", GetType(T), x)
End Sub
```

والآن في حالة تطبيقنا لنفس المثال السابق ، سوف يكون الناتج كالتالي:

```
file:///C:/Users/Ahmed/Documents/Visual Studio 2008/Projects/ConsoleApplication4/ConsoleApplic...
Print As System.String: Ahmed
Print As System.Int32: 12
Print As System.Int64: 10000000000000
```

الصورة 12. 2. ناتج تنفيذ العمليات

1. 1. Structure and Class Generics

في الدرس السابق تعرفنا على ال methods التي تطبق ال Generics، ايضاً ال Classes وال Structs يمكنها تطبيق هذا المفهوم لبعض البيانات بها ، سنفترض مثال ال class وما ينطبق على ال `class` ينطبق على ال `struct`.

لنفترض فئة تحتوي على البضائع ، وحيث ان لدينا نوعين من البضائع يحتوي النوع الأول على الاسم وهي تلك البضائع التي يتم تصنيعها في المحل (انوع من الجبن او الزبادي) وبضائع

أخرى جاهزة تحتوي على الباركود ، بهذه الطريقة نستنتج ان حقل Identify قد يحتوي على اسم في بعض البضائع ورقم في بضائع أخرى.

الحل التقليدي كان ليعتبر وجود فئتين منفصلتين، أو في أحسن الظروف وضع قيمة منطقية لتحديد اذا ما كنت ساستخدم حقل ال name ام حقل ال number، لكن مع ال Generics تم تغيير هذا المفهوم ، في هذه الحالة ستصبح الفئة بالشكل التالي:

كود C#

```
public class product<T>
{
    private T Identify;
    public product(T val)
    {
        Identify = val;
        Console.WriteLine(Identify);
    }
}
```

كود VB.NET

```
Public Class product(Of T)
    Private Identify As T

    Public Sub New(ByVal val As T)
        Identify = val
        Console.WriteLine(Identify)
    End Sub
End Class
```

والآن يمكنني تعريف كائن من النوع الأول بالشكل التالي:

كود C#

```
product<int> prd = new product<int>(1001);
```

كود VB.NET

```
Dim prd As New product(Of Integer)(1001)
```

أو من النوع الثاني بالشكل التالي:

كود C#

```
product<string> prd2 = new product<string>("Some Product");
```

كود	VB.NET

```
Dim prd2 As New product(Of String)("Some Product")
```

1 .2. Generic Collection

كما تعرفنا في الدرس السابق ان بامكاننا عمل Generic Class، اصبح الدور الآن على عمل Generic Collection، سنقوم بعمل فئة تستقبل عدة فئات ويقوم بعدة عمليات مثل Add, Delete ... etc، كما تعلمنا سابقاً ، هذا مثال على Collection ما:

كود	C#

```
public class MyCollection<T>
{
    private List<T> myList = new List<T>();
    public T GetItem(int pos)
    {
        return myList[pos];
    }
    public void AddItem(T it)
    {
        myList.Add(it);
    }
    public void ClearCars()
    {
        myList.Clear();
    }
}
```

كود	VB.NET

```
Public Class MyCollection(Of T)
    Private myList As New List(Of T)()
    Public Function GetItem(ByVal pos As Integer) As T
        Return myList(pos)
    End Function
    Public Sub AddItem(ByVal it As T)
        myList.Add(it)
    End Sub
    Public Sub ClearCars()
        myList.Clear()
    End Sub
End Class
```

طبعاً بامكانك اضافة الكثير من الخصائص وربما بعض الخصائص الغير متوفرة اصلاً للبحث وخلافه ، هذا ما يسمى بال Generic Collection حيث يمكن في مراحل لاحقة الاعتماد عليه بدل ال Colelctions العادية حيث يمكن ان يستقبل Car او .... Employee إلى غير ذلك من الخيارات ، بالشكل التالي مثلاً :

كود	C#

```csharp
Identify = val;
Console.WriteLine(Identify);
MyCollection<Car> myCars = new MyCollection<Car>();
myCars.AddItem(temp);
MyCollection<Employee> myCars = new MyCollection<Employee>();
myCars.AddItem(tempEmployee);
```

كود	VB.NET

```vb
Dim myCars As New MyCollection(Of Car)()
myCars.AddItem(temp)
Dim myCars As New MyCollection(Of Employee)()
myCars.AddItem(tempEmployee)
```

أو حتى ارقام:

كود	C#

```csharp
MyCollection<int> myCars = new MyCollection<int>();
myCars.AddItem(12);
```

كود	VB.NET

```vb
Dim myCars As New MyCollection(Of Integer)()
myCars.AddItem(12)
```

هنا ظهرت لنا مشكلة تتعلق بكون ال Generic Collection الخاص بنا يحتوي على خصائص مثل name. لعرضها ، وهذا ما لا يتوفر مثلاً لكل الفئات او للارقام مثلاً ، من هنا جاءت لنا الكلمة المحجوزة where والتي سنتحدث عنها في الدرس القادم.

1. 3. استخدام where T

يتيح لنا هذا الشرط في عمل ال Generic Collection ان لا نقبل سوى بعض الفئات ، يمكننا وضع الشروط بأحد الطرق التالية:

الشرط	المعنى
where T : struct	بحيث لا يقبل سوى متغيرات من النوع ValueType
where T : class	لا يقبل سوى من النوع RefernceType
where T :new()	لا بد ان يحتوي على Constructor
where T : ClassName where T : IInterfaceName	لا بد ان يكون مشتق من class معين او يطبق interface معين.

الجدول 12. 1. استخدام الشروط مع ال Generics

بهذه الطريقة يمكننا دمج عدة شروط مع بعضها ، لنرى المثال التالي سوية:

كود C#
```
public class Example <T> where T : class, IComparable, new()
```

كود VB.NET
```
Public Class Example(Of T As {Class, IComparable, New})
```

هذا يعني ان T لا بد ان تكون reference ، تطبق الواجهة IComparable وتحتوي على Constructor.

ايضاً المثال التالي:

كود C#
```
public class Example<K, T> where K : class, new()
where T : IComparable<T>
```

كود VB.NET
```
Public Class Example(Of K As {Class, New}, T As IComparable(Of T))
```

هذا يعني ان K لا بد ان تكون Reference ولها constructor فيما T لا بد ان تطبق الواجهة IComparable .

والآن ، اصبح بامكانك التحكم بعض الشيء في T بدلاً من جعلها مفتوحة للجميع.

2. ال Delegates

نواصل رحلتنا مع ال net programming. advanced، موعدنا هذه المرة مع الDelegates.

قبل ان نشرح تركيبها وكيفية التعامل معها ، سنقوم بشرح لماذا نقوم باستخدامها.

لنفترض عدة دوال تستقبل int وتعيد int ايضاً خاصة بعمليات التحويل ، وهي على الشكل التالي:

كود	C#

```
public int ConvertEGToD(int EG)
{
    return EG * 5.45;
}
public int ConvertRSToD(int RS)
{
    return RS * 3.75;
}
public int ConvertEGToRS(int EG)
{
    return EG * 1.45;
}
public int ConvertDToRS(int D)
{
    return D * 3.75;
}
public int ConvertDToEG(int D)
{
    return D / 5.45;
}
public int ConvertRSToEG(int RS)
{
    return RS / 1.45;
}
```

كود	VB.NET

```
Public Function ConvertEGToD(ByVal EG As Integer) As Integer
    Return EG * 5.45
End Function
Public Function ConvertRSToD(ByVal RS As Integer) As Integer
    Return RS * 3.75
End Function
Public Function ConvertEGToRS(ByVal EG As Integer) As Integer
    Return EG * 1.45
End Function
Public Function ConvertDToRS(ByVal D As Integer) As Integer
    Return D * 3.75
End Function
Public Function ConvertDToEG(ByVal D As Integer) As Integer
    Return D / 5.45
End Function
Public Function ConvertRSToEG(ByVal RS As Integer) As Integer
    Return RS / 1.45
End Function
```

كما لاحظت فعلاً، فهي عدة دوال تستخدم للتحويلات المختلفة بين ثلاث عملات، الجنية المصري والريال السعودي والدولار الأمريكي.

طبعاً يمكننا عملهم في دالة واحدة وارسال متغير يمثل رقم التحويل ، لكننا لن نستطيع عمل ذلك مثلاً مع دوال اكبر ومختلفة ، لذا فهذا المثال للتوضيح.

الآن في برنامجنا سنقرأ البيانات من المستخدم ، ومن ثم نستخدم جملة شرط if او switch لتحديد اي دالة سنقوم بارسال البيانات إليها ، هذا مختصر للكود المكتوب:

كود	C#

```
if (Operation == 0)
        result = ConvertDToEG(userInput);
    else if(Operation == 1)
        result = ConvertRSToD(userInput);
```

كود	VB.NET

```
If Operation = 0 Then
   result = ConvertDToEG(userInput)
ElseIf Operation = 1 Then
   result = ConvertRSToD(userInput)
End If
```

ولكن لنفترض اننا فقط الآن نود معرفة نوع العملية دون تنفيذها وعرضها للمستخدم حيث سننفذها لاحقاً ، هذا يعني اننا سنقوم بذات الاختبار مرتين ، مرة للعرض على المستخدم ومرة

لتنفيذ العملية ، ايضاً ألا تتفق معي في انك قد تحتاج لتنفيذ العملية مرتين في مكانين مختلفين، وما دمت لا تحتفظ سوى برقم ال Operation إذا ستضطر لعمل جمل الشرط مرة أخرى.

من أجل هذا وجدت ال Delegates.

2. 1. تعريف ال Delegates

لو اردنا تعريف Delegates لمجموعة دوال ، اول ما نحتاج إليه ان تكون هذه الدوال من نفس عدد البارميترات . ايضاً لها نفس ال input وال output ، لذا سيكون ال Delegate الخاص بدوالنا بالشكل التالي:

كود C#

```csharp
public delegate int myDelegate(int value);
```

كود VB.NET

```vbnet
Public Delegate Function myDelegate(ByVal value As Integer) As Integer
```

والآن كل ما علي في جملي الشرطية ان احدد لهذا التفويض - إن صحت الترجمة - الدالة المسؤول عنها ، لذا ستكون شروطنا المختصرة بالشكل التالي

كود C#

```csharp
if (Operation == 0)
{
    myDelegate aDelegate = new myDelegate(ConvertDToEG);
    result = aDelegate(userInput);
}
else if (Operation == 1)
{
    myDelegate aDelegate = new myDelegate(ConvertRSToD);
    result = aDelegate(userInput);
}
```

كود VB.NET

```vbnet
If Operation = 0 Then
    Dim aDelegate As New myDelegate(ConvertDToEG)
    result = aDelegate(userInput)
ElseIf Operation = 1 Then
    Dim aDelegate As New myDelegate(ConvertRSToD)
    result = aDelegate(userInput)
End If
```

ايضا بامكانك تأخير الشرط الأخير الذي يقوم بتنفيذ ال Delegate للنهاية ، او تكرار استخدامها مرة أخرى ، حيث ان ال aDelegate قد اصبحت تعرف اي دالة تختص بتنفيذ هذه العملية الآن ، لذا يمكنك كتابة الكود بالشكل التالي :

كود C#

```
if (Operation == 0)
{
    myDelegate aDelegate = new myDelegate(ConvertDToEG);
}
else if (Operation == 1)
{
    myDelegate aDelegate = new myDelegate(ConvertRSToD);
}
// هنا الأوامر بعض نفذ
result = aDelegate(userInput);
```

كود VB.NET

```
If Operation = 0 Then
    Dim aDelegate As New myDelegate(ConvertDToEG)
ElseIf Operation = 1 Then
    Dim aDelegate As New myDelegate(ConvertRSToD)
End If
    'هنا الأوامر بعض نفذ
result = aDelegate(userInput)
```

وهكذا ، يمكنك تأخير استدعاء الدالة حتى تنتهي من تنفيذ أي أوامر مرتبطة دون ان تقلق من أنك ستعيد التأكد مرة أخرى ، هذه واحدة .

والثانية يمكنك عمل التأكد في دالة منفصلة واعادة ال Delegate جاهز للتنفيذ ، هذا يفيد كثيراً في حالة العمل على شكل مجموعة .

2 .2. الأحداث Events

لو جربنا الآن ان نقوم بعمل حدث معين لأي موظف عندنا (مرض ... الخ أو لاي سيارة (حادث اصدام ... الخ) كنا نقوم بذلك سابقاً عن طريق Delegates باسم Event مثلاً ، ونقوم

بتمرير الدالة الخاصة بالحدث له ، وإذا كنت قد توسعت في ال Delegates فأنت قادر على معرفة انك تستطيع عمل List بالاحداث التي تم تنفيذها على هذا ال Delegate.

الآن سنتعلم الأمر بطريقة جديدة عن طريق الكلمة المحجوزة Event ...

سنعرف في البداية Delegate مسؤول عن كافة الأحداث التي تحصل للسيارة مثلاً لعمل Delegate يستقبل نص الرسالة المطلوبة :

كود C#
```
public delegate void CarEventHandler(string msg);
```

كود VB.NET
```
Public Delegate Sub CarEventHandler(ByVal msg As String)
```

والآن سنقوم بتعريف بعض الاحداث:

كود C#
```
public event CarEventHandler Exploded;
public event CarEventHandler Damaged;
```

كود VB.NET
```
Public Event Exploded As CarEventHandler
Public Event Damaged As CarEventHandler
```

الآن يمكنك ببساطة من خلال الكود تنفيذ اي حدث فيهم بالشكل التالي:

كود C#
```
Damaged("my car");
```

كود VB.NET
```
Damaged("my car")
```

وأيضاً يمكنك اختبار اي event == null ام لا لمعرفة إذا كان تم اطلاقه قبل ذلك أم لا، لمعرفة هل السيارة مثلاً تم تدميرها أم لا.

آخر نقطة ، لاضافة دالة الحدث إلى الكائن :

كود C#
```
Car.EngineHandler d = new Car.CarEventHandler(CarExploded);
```

كود VB.NET

```
Dim d As Car.EngineHandler = New Car.CarEventHandler(CarExploded)
```

حيث يتم تعريف الدالة CarExploded لتنفيذ المهمة المطلوبة وهي اظهار رسالة بناء على النص المرسل والذي تم ارساله في الحدث Damaged ، بالشكل التالي مثلاً :

كود C#

```
public void CarExploded(string msg)
{
       Console.WriteLine(msg);
}
```

كود VB.NET

```
Public Sub CarExploded(ByVal msg As String)
    Console.WriteLine(msg)
End Sub
```

3. ال Anonymous Methods فقط في C#

إذا كنت قد استوعبت الدرس السابق عن ال Delegates فأنت تدرك أن بامكانك استدعاء دالة باستخدام متغير من النوع Delegate .

وبعد الدرس الخاص ب Events اصبحت تدرك ان بامكانك اضافة اسم دالة ليمثل الحدث الذي قمت ببرمجته بالشكل التالي مثلاً:

كود C#

```
t.SomeEvent += new SomeDelegate(MyEventHandler);
```

والآن لنفترض انك لا تريد استدعاء الدالة سوى في هذا المكان فقط ، لذا سيكون من المكلف تعريف الدالة ومن ثم استدعاءها في Delegate ، هنا يظهر لنا ما يعرف باسم Anonymous Methods حيث بامكانك تعريف الدالة وسط الكود.

لنرى الكود التالي مثلاً:

كود C#

```
t.SomeEvent += delegate{
                   Console.WriteLine("Some Text");
               }
```

نعم هذا صحيح ولكن فقط مع #C. كما ترى اصبح الآن بامكانك تعريف الدالة في موقع استخدامها فقط ، يمكن ايضاً ان تكون الدالة تستقبل عدة بارميترات بالشكل التالي مثلاً:

كود C#

```
t.SomeEvent += delegate(object sender, CarEventArgs e)
                {
                    Console.WriteLine("Some Text {0}",e.msg);
                }
```

ايضاً بامكان ال Anonymous method ان تصل إلى المتغيرات الموجودة في الدالة التي تم تعريفها فيها.

4. استنتاج أنواع المتغيرات

في ايام الفيجوال بيسك 6 ، كنا قادرين على تعريف متغير دون تحديد نوعه بالشكل التالي:

كود VB.NET

```
Dim x
x = 10
```

في الواقع كان الفيجوال بيسك يقوم بتعريفها مبدئياً من النوع Object، في حين كان مثل هذا الامر ممنوعاً في اللغات التي تتبع عائلة السي.

مع net 2008. ، اصبح بامكان السي شارب تعريف متغير بدون الحاجة إلى تحديد نوعه بالشكل التالي مثلاً:

كود C#

```
var x = 2.3 // double
```

كود VB.NET

```
Dim x = 2.3 ' double
```

الهدف من هذه العملية هو تعريف متغير قادر على حمل اي نوع من البيانات ، لكن كن حذراً ، فلن يمكنك مثلاً تعريف var في الفئة مباشرة ، او في بارميترات الدالة أو في ال return value لها ، أخيراً لا يمكن لل var أن يحمل قيمة null.

المثال التالي يجمع الأخطاء التي لا يمكن استخدام var فيها:

كود	C#

```
class classname
{
    // لا يمكن تعريفة في الفئة مباشرة
    private var varInt = 10;
    // لا يمكن التعريف كبارميتر او ك return value .
    var functionname(var x, var y){  }
    void somefunction()
    {
        // لا يمكن ان يحمل قيمة null
        var varNull=null;
        // وحتى بهذه الطريقة لا يمكن ان يحمل قيمة null
        var? varNullable = 12;
        // لا بد من تعيين قيمة
        var m;
    }
}
```

5. الدوال الممتدة Extension Methods

ال Extension Methods واحدة من خواص net 2008. الجديدة ، تتيح لك هذه الخاصية التعديل على فئات موجودة مسبقاً واضافة دالة أو دوال جديدة.

لنفترض اننا نريد اضافة دالة للفئة string لتقوم بالتأكد من صحة البريد الالكتروني ، سنقوم بعمل دالة تستخدم Regex بالشكل التالي مثلاً:

كود	C#

```
public static class Extensions
{
    public static bool IsValidEmailAddress(this string s)
    {
        Regex regex = new Regex(@"^[\w-\.]+@([\w-]+\.)+[\w-]{2,4}$");
        return regex.IsMatch(s);
    }
}
```

VB.NET	كود

```
Imports System.Runtime.CompilerServices
Module StringExtensions
    <Extension()>_
    Public Function IsValidEmailAddress(ByVal s As String) As Boolean
        Dim regex As New Regex("\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*")
        Return regex.IsMatch(s)
    End Function
End Module
```

والآن ، يمكننا بكل بساطة تعريف متغير string بالشكل التالي:

C#	كود

```
string mailExample = "email@mail. com";
MessageBox.Show(mailExample.IsValidEmailAddress.toString());
```

VB.NET	كود

```
Dim mailExample As String = "email@mail. com"
MessageBox.Show(mailExample.IsValidEmailAddress.toString())
```

6. Automatic Properties

سابقاً ومن اجل انشاء Properties كنا نقوم بتعريف دوال Set و Get للقراءة والكتابة بالشكل التالي مثلاً:

C#	كود

```
public class myclass
{
    private string _name;
    public string name
    {
        get {return _name;}
        set {_name=value;}
    }
}
```

كود	VB.NET

```
Public Class [myclass]
    Private _name As String
    Public Property name() As String
        Get
            Return _name
        End Get
        Set(ByVal value As String)
            _name = value
        End Set
    End Property
End Class
```

الآن اصبح بامكانك كتابة الكود بالشكل التالي:

كود	C#

```
public class myclass
{
    public string name{get; set;}
}
```

كود	VB.NET

```
Public Class [myclass]
    Public Property name() As String
        Get
        End Get
        Set(ByVal value As String)
        End Set
    End Property
End Class
```

7. تعابير لامدا Lamda Expressions

كنا قد تحدثنا في موضوع سابق عن Anonymous Methods التي تمكننا من كتابة كود الدالة في مكان استدعاءها ما دمنا لن نستدعيها سوى مرة واحدة فقط ، لنفترض مصفوفة نقوم فيها بالبحث عن الأعداد التي تقبل القسمة على 2:

كود	C#

```
static void TraditionalDelegateSyntax()
{
    List<int> list = new List<int>();
    list.AddRange(new int[] { 20, 1, 4, 8, 9, 44 });
    Predicate<int> callback = new Predicate<int>(IsEvenNumber);
    List<int> evenNumbers = list.FindAll(callback);
    Console.WriteLine("Here are your even numbers:");
    foreach (int evenNumber in evenNumbers)
    {
        Console.Write("{0}\t", evenNumber);
    }
}

static bool IsEvenNumber(int i)
{
    return (i % 2) == 0;
}
```

كود	VB.NET

```
Private Shared Sub TraditionalDelegateSyntax()
    Dim list As New List(Of Integer)()
    list.AddRange(New Integer() {20, 1, 4, 8, 9, 44})
    Dim callback As New Predicate(Of Integer)(AddressOf IsEvenNumber)
    Dim evenNumbers As List(Of Integer) = list.FindAll(callback)
    Console.WriteLine("Here are your even numbers:")
    For Each evenNumber As Integer In evenNumbers
        Console.Write("{0}" & Chr(9) & "", evenNumber)
    Next

End Sub
Private Shared Function IsEvenNumber(ByVal i As Integer) As Boolean
    Return (i Mod 2) = 0
End Function
```

مع استخدامنا لل Anonymous Methods من C# 2008 اصبح باستطاعتنا كتابتها بالشكل التالي كما تعرف:

كود	C#

```
static void TraditionalDelegateSyntax()
{
    List<int> list = new List<int>();
    list.AddRange(new int[] { 20, 1, 4, 8, 9, 44 });
    Predicate<int> callback = new Predicate<int>(IsEvenNumber);
    List<int> evenNumbers = list.FindAll(callback);
    Console.WriteLine("Here are your even numbers:");
    foreach (int evenNumber in evenNumbers)
    {
        Console.Write("{0}\t", evenNumber);
    }
}

static bool IsEvenNumber(int i)
{
    return (i % 2) == 0;
}
```

كود	VB.NET

```
Dim list As New List(Of Integer)()
list.AddRange(New Integer() {20, 1, 4, 8, 9, 44})

Dim evenNumbers As List(Of Integer) = list.FindAll(AddressOf
ConvertedAnonymousMethod1)
Console.WriteLine("Here are your even numbers:")
For Each evenNumber As Integer In evenNumbers
    Console.Write("{0}" & Chr(9) & "", evenNumber)
Next
```

الجديد في net 2008. هو استخدام ما يعرف باسم Lambda Expressions، والتي يمكن كتابتها بالشكل التالي:

كود

```
X => f(X)
```

لذا ستكون الدالة الخاصة بنا والتي تعيد True في حالة i%2==0 بالشكل التالي:

كود	C#

```
(int i) => (i % 2) == 0;
```

كود	VB.NET

```
Function(i As Integer) (i Mod 2) = 0
```

حيث ان ال i هي البارميتر ، ونوعه int ، يمكنك حتى الاستغناء عن تعريف نوع البارميتر لإن Lambda سوف تتعرف عليه تلقائياً ، لذا سيكون الكود الكامل بالشكل التالي:

كود C#

```
List<int> list = new List<int>();
list.AddRange(new int[] { 20, 1, 4, 8, 9, 44 });
List<int> evenNumbers = list.FindAll(i => (i % 2) == 0);
Console.WriteLine("Here are your even numbers:");
foreach (int evenNumber in evenNumbers)
{
    Console.Write("{0}\t", evenNumber);
}
```

كود VB.NET

```
Dim list As New List(Of Integer)()
list.AddRange(New Integer() {20, 1, 4, 8, 9, 44})
Dim evenNumbers As List(Of Integer) = list.FindAll(Function(i As ) (i Mod 2) =
0)
Console.WriteLine("Here are your even numbers:")
For Each evenNumber As Integer In evenNumbers
    Console.Write("{0}" & vbTab, evenNumber)
Next
```

:return value

يمكننا ليس فقط اعادة قيمة واحدة أو true , false فقط ، يمكنك الاطلاع على المثال التالي:

كود C#

```
List<int> evenNumbers = list.FindAll((i) =>
{
    Console.WriteLine("value of i is currently: {0}", i);
    bool isEven = ((i % 2) == 0);
    return isEven;
});
```

ايضاً يمكننا تمرير اكثر من بارميتر ، وذلك حسب الدالة...

مواصفات Lambda:

- لا تملك Lambda خاصية الإسم .

- لا يمكن استخدام بعض مبادئ ال OOP مثل Overloads أو Overrides.
- لا تستخدم قسم As لتحديد نوع القيمة المعادة وبدلا من ذلك يتم استنتاج المتغيرات تلقائياً .
- داخل التعبير لا بد من وجود كود وليس تعريف لشيء آخر ، ويمكن استدعاء دالة أيضاً .
- لا يتم استخدام Return بل يتم اعادة قيم الدالة مباشرة .
- لا يوجد End أو {} في Lambda.
- لا يمكن استخدام Optinal في VB.
- لا يمكن استخدام Generic.

8. صيغ انشاء الكائنات Object Initializer Syntax

لن نطيل كثيراً في هذا الدرس ، فقط سنتعرض لعدة امثلة سريعة عن فئة الموظفين التي تحتوي خصائص الاسم والعمر والمرتب.

المثال الأول ما قبل net 2008.:

كود C#

```
public class Employee
{
    private string _name;
    private string _age;
    private string _salary;
    public string name
    {
        get { return _name; }
        set { _name = value; }
    }
    public string age
    {
        get { return _age; }
        set { _age = value; }
    }
    public string salary
    {
        get { return _salary; }
        set { _salary = value; }
    }
}
```

كود	VB.NET

```
Public Class Employee
    Private _name As String
    Private _age As String
    Private _salary As String
    Public Property name() As String
        Get
            Return _name
        End Get
        Set(ByVal value As String)
            _name = value
        End Set
    End Property
    Public Property age() As String
        Get
            Return _age
        End Get
        Set(ByVal value As String)
            _age = value
        End Set
    End Property
    Public Property salary() As String
        Get
            Return _salary
        End Get
        Set(ByVal value As String)
            _salary = value
        End Set
    End Property
End Class
```

المثال الثاني مع net 2008. وباستخدام automatic property التي تعلمناها في درس سابق:

كود	VB.NET

```
Public Class Employee
    Public Property name() As String
        Get: End Get
        Set(ByVal value As String)
        End Set
    End Property
    Public Property age() As String
        Get: End Get
        Set(ByVal value As String)
        End Set
    End Property
    Public Property salary() As String
        Get:End Get
        Set(ByVal value As String)
        End Set
    End Property
End Class
```

كود C#

```
public class Employee
{
    public string name { get; set; }
    public string age { get; set; }
    public string salary { get; set; }
}
```

المثال الثالث ومع استخدام Object Initializer Syntax الجديد من2008 net.، نريد الوصول إلى الخصائص التي سبق لنا تعريفها بأحد الطرق السابقة ، في السابق كنا نكتب كود بالشكل التالي:

كود C#

```
Employee e = new Employee();
e.Name="Ahmed";
e.Age=15;
e.salary=6000;
```

كود VB.NET

```
Dim e As Employee = New Employee()
e.Name = "Ahmed"
e.Age = 15
e.salary = 6000
```

الجديد هنا:

كود C#

```
Employee e = new Employee { Name = "Ahmed", Age = 15, salary = 6000 };
```

كود VB.NET

```
Dim e As Employee = New Employee With {.Name="Ahmed" ,.Age=15,.Salary=1500}
```

9. الأنواع المجهولة Anonymous Types

نواصل في هذا الدرس شرح الخصائص الجديدة في2008 net.، في هذا الدرس سوف نتعرف على Anonymous Types .

لا تنسى اننا تعلمنا ان بامكاننا تعريف متغيرات مجهولة باستخدام الكلمة var في السي شارب ، اما الآن فلم تعد متغيرات مجهولة الهوية فقط بل بامكانك تعريف انواع مجهولة ايضاً ، لن اطيل كثيراً في المقدمة ، انظر إلى الكود التالي :

كود C#

```
var e = new { Name = "Ahmed", Age = 20 };
```

كود VB.NET

```
Dim e As Employee = New Employee With {.Name="Ahmed" ,.Age=15,.Salary=1500}
```

الآن اصبح بامكانك قراءة e.Name و e.Age بدون تعريف الفئة اصلاً، فعلياً ال Anonymous Types يتم اسناده مباشرة إلى System.Object لذا فهو يحتوي على خصائصه الاساسية.

ماذا استفيد ؟

في LINQ يمكننا تعريف فئة ترتبط بناتج جملة الاستعلام ، لنفترض اننا نود قراءة ناتج جملة استعلام و لا نعلم تحديداً شكل الناتج ، يتم ذلك بالشكل التالي :

كود C#

```
var result = from emp in employee select new { emp.Name, emp.Salary };
```

كود VB.NET

```
Dim namePriceQuery = From emp In employee Select emp.Name, emp.Salary
```

الخصائص المفتاحية Key Properties

تختلف الخصائص المفتاحية عن العادية بعدة أمور

- تستخدم الخصائص المفتاحية فقط لمقارنة المساواة بين نوعين مجهولين

- لايمكن تغيير قيم الخصائص المفتاحية فهي دائما للقراءة فقط

- فقط الخصائص المفتاحية يتم تضمينها ضمن الـ Hash Code الذي يولده المترجم من أجل الأنواع المجهولة .

المساواة Equality

تكون متغيرات الأنواع المجهولة متساوية عندما تكون متغيرات لنفس النوع المجهول ويقوم المعالج بمعاملة متغيرين كمتغيرين من نفس النوع إذا توفرت فيهما الشروط التالية :

- تم التصريح عنهما في نفس المدى Scope .

- تمتلك خصائصهما نفس الاسم والنوع وتم التصريح عنها بنفس الترتيب وتكون مقارنة الأسماء غير حساسة لحالة الأحرف .

- نفس الخصائص فيها محددة كخصائص أساسية .

- يمتلك كل نوع خاصية أساسية واحدة على الأقل .

والتصريح عن نوع مجهول لايمتلك أي خاصية مفتاحية يكون مساويا لنفسه فقط .

كما لا تنس أنه لا يمكن تغيير قيم الخصائص المفتاحية ...

10. Partial Methods

باختصار شديد ، لم تعد الآن مجبراً على وضع الفئة في منطقة واحدة ، بل ان بامكانك كتابة الفئة في اكثر من موضع ، حيث يكفي ان تستخدم الكلمة Partial في اسم الفئة لتدل على ان هذه ليس فئة جديدة بل هو يتبع فئة معرفة في مكان آخر ، بالشكل التالي مثلاً:

كود	C#

```
partial class Car
{
}
```

كود　VB.NET

```
Partial Class Car

End Class
```

كان هذا مع عصر net 2005. ، ولكن بعد ذلك ظهر مؤخراً ما يعرف باسم Partial Methods، حيث لم تعد مضطراً لكتابة الدالة في مكان واحد ايضاً ، يتم ذلك بالشكل التالي :

كود　C#

```
partial void methodname(string parm)
{

}
```

كود　VB.NET

```
Partial Private Sub methodname(ByVal parm As String)


End Sub
```

طبعاً كما تلاحظ ، افادنا هذا الموضوع كثيراً في تطوير خصائص الفئات الاساسية ودوالها بدون الحاجة إلى تعريف نسخة جديدة منها.

11. Garbage Collector

خلال دروسنا السابقة كنا نعرف متغيرات وفئات .. الخ ، ولكننا لم نسأل نفسنا كيف يتم تخزينها في الذاكرة ومتى يتم حذفها ، هذه الاسئلة سوف نجيب عليها في هذا الدرس من خلال مفاهيم ال Object lifetime وال GC.

عند تعريفك لكائن من فئة OBJECT FROM CLASS فإنك بالواقع تصبح ممسكاً ب Refernce يشير إلى هذا الكائن الموجود في ال heap ، هذا ال reference يوجد ايضاً في stack خاص بالبرنامج.

بعد اغلاق البرنامج او انتهاء الدالة يتم حذف ال reference من ال stack ، سيكون التساؤل الطبيعي هو انه وفي هذه الحالة سوف تمتلئ الذاكرة بمئات الكائنات التي لا تجد من يشير لها ، يريحك ال garbage collector والذي يرمز له اختصاراً GC من هذا التساؤل حيث يقوم بحذف

الفئات غير المستخدمة ، او في الحقيقة فهو يقوم بحذف جميع الفئات التي لم يعد بامكانك الوصول إليها من داخل البرنامج.

في ايام ال ++C كان من المفترض ان تقوم بحذف متغيراتك الغير مستخدمة اول بأول، اما الآن مع GC فاصبحت جل المهام يتم تنفيذها دون ان تشعر.

الحالة الاكثر شيوعاً هي ان تقوم بعمل null= ، في الواقع هذا لا يعطي اشارة مباشرة لل GC لحذف كائنك من الذاكرة ، لكنه سيتم حذفه في لحظة ما لا يمكنك التحكم بها ، عندما تمتلئ الذاكرة مثلاً.

11 .1. الفئة GC

تحتوي الفئة GC على عدد من الداول التي تمكنك من التعامل المباشر معها ، هذه صورة من كتاب Pro Csharp 2008 الدوال واستخداماتها:

System.GC Member	Meaning in Life
AddMemoryPressure() RemoveMemoryPressure()	Allow you to specify a numerical value that represents the calling object's "urgency level" regarding the garbage collection process. Be aware that these methods should alter pressure *in tandem* and thus never remove more pressure than the total amount you have added.
Collect()	Forces the GC to perform a garbage collection. This method has been overloaded to specify a generation to collect, as well as the mode of collection (via the GCCollectionMode enumeration).
CollectionCount()	Returns a numerical value representing how many times a given generation has been swept.
GetGeneration()	Returns the generation to which an object currently belongs.
GetTotalMemory()	Returns the estimated amount of memory (in bytes) currently allocated on the managed heap. The Boolean parameter specifies whether the call should wait for garbage collection to occur before returning.
MaxGeneration	Returns the maximum of generations supported on the target system. Under Microsoft's .NET 3.5, there are three possible generations (0, 1, and 2).
SuppressFinalize()	Sets a flag indicating that the specified object should not have its Finalize() method called.
WaitForPendingFinalizers()	Suspends the current thread until all finalizable objects have been finalized. This method is typically called directly after invoking GC.Collect().

الصورة 12 .1. دوال الفئة GC

:Finalize()

يمكنك هذا الحدث من اقتناص وقت حذف الكائن ، يمكنك عمل overriding له وتنفيذ بعض المهام قبل تدمير الكائن.

يتم ذلك بالشكل التالي:

كود C#

```
class example
{
    ~example()
    {
        Console.Beep();
    }
}
```

كود VB.NET

```
Class example
    Protected Overrides Sub Finalize()
        Try
            Console.Beep()
        Finally
            MyBase.Finalize()
        End Try
    End Sub
End Class
```

12. Operator Overloading

في الانواع الرئيسية لنا ، نستخدم المعاملات المختلفة لتنفيذ عمليات على الفئات المشتقة منها ، فمثلاً المتغير من نوع Integer يفهم المعامل + على انه جمع ، - على انه طرح ... الخ.

ايضاً المتغيرات من نوع String تفهم المتغير + مثلاً على انه لدمج نصين ، وهكذا.

الآن لو قمنا بعمل Structure من نوع Point بالشكل التالي مثلاً:

كود C#

```
public struct Point
{
    private int x, y;
    public Point(int xPos, int yPos)
    {
        x = xPos;
        y = yPos;
    }
}
```

كود	VB.NET

```
Public Structure Point
    Private x As Integer, y As Integer
    Public Sub New(ByVal xPos As Integer, ByVal yPos As Integer)
        x = xPos
        y = yPos
    End Sub
End Structure
```

الآن جرب تعريف عدة نقاط ، واستخدام المعامل + او - لجمع وطرح النقاط ، ما تتوقعه ان تشاهد الناتج .

عبارة عن طرح ال x في النقطة الثانية من الأولى وكذا ال y ، أو جمعهما معاً ، إلا انك في الواقع لن تحصل سوى على رسالة خطأ تفيدك بأن structure المسمى Point لا يدعم معاملات الجمع والطرح .

الآن سنقوم باضافة معاملات جمع وطرح إلى ال structure السابق:

كود	C#

```
public struct Point
{
    private int x, y;
    public Point(int xPos, int yPos)
    {
        x = xPos;
        y = yPos;
    }

    public static Point operator +(Point p1, Point p2)
    { return new Point(p1.x + p2.x, p1.y + p2.y); }
    public static Point operator -(Point p1, Point p2)
    { return new Point(p1.x - p2.x, p1.y - p2.y); }
}
```

كود	VB.NET

```
Public Structure Point
    Private x As Integer, y As Integer
  Public Sub New(ByVal xPos As Integer, ByVal yPos As Integer)
        x = xPos
        y = yPos
    End Sub

  Public Shared Operator +(ByVal p1 As Point, ByVal p2 As Point) As Point
        Return New Point(p1.x + p2.x, p1.y + p2.y)
    End Operator

  Public Shared Operator -(ByVal p1 As Point, ByVal p2 As Point) As Point
        Return New Point(p1.x - p2.x, p1.y - p2.y)
    End Operator
End Structure
```

الآن يمكنك كتابة كود بالشكل التالي:

كود	C#

```
Point p3 = p1 + p2;
```

كود	VB.NET

```
Dim p3 As Point = p1 + p2
```

ستحصل على النتيجة الصحيحة لعملية جمع ال Points.

أيضاً يمكنك كتابة كود كالتالي مباشرة:

كود	C#

```
p2-=p1;
```

كود	VB.NET

```
p2-=p1
```

آخر ما سنتعرف عليه في هذا الجزء من الدرس ، انك لن تكون قادراً سوى على تعريف static function من اجل عمل overloading لأي معامل.

ليست معاملات الجمع والطرح فقط من يمكن عمل overloading، بل يمكنك عمل ذلك لأي نوع من المعاملات ، فمثلاً ++ و -- في السي شارب فقط:

C#	كود

```
public static Point operator ++(Point p1)
{
    return new Point(p1.x + 1, p1.y + 1);
}
public static Point operator --(Point p1)
{
    return new Point(p1.x - 1, p1.y - 1);
}
```

نفس الأمر بالنسبة لدوال المساواة وعمل == أو =! حقيقية مثل ما تعلمنا سابقاً مع عمل overriding للدالة Equals، سنعتمد هنا على هذه الدالة بالشكل التالي:

C#	كود

```
public override bool Equals(object o)
{
    return o.ToString() == this.ToString();
}

public static bool operator ==(Point p1, Point p2)
{
    return p1.Equals(p2);
}

public static bool operator !=(Point p1, Point p2)
{
    return !p1.Equals(p2);
}
```

VB.NET	كود

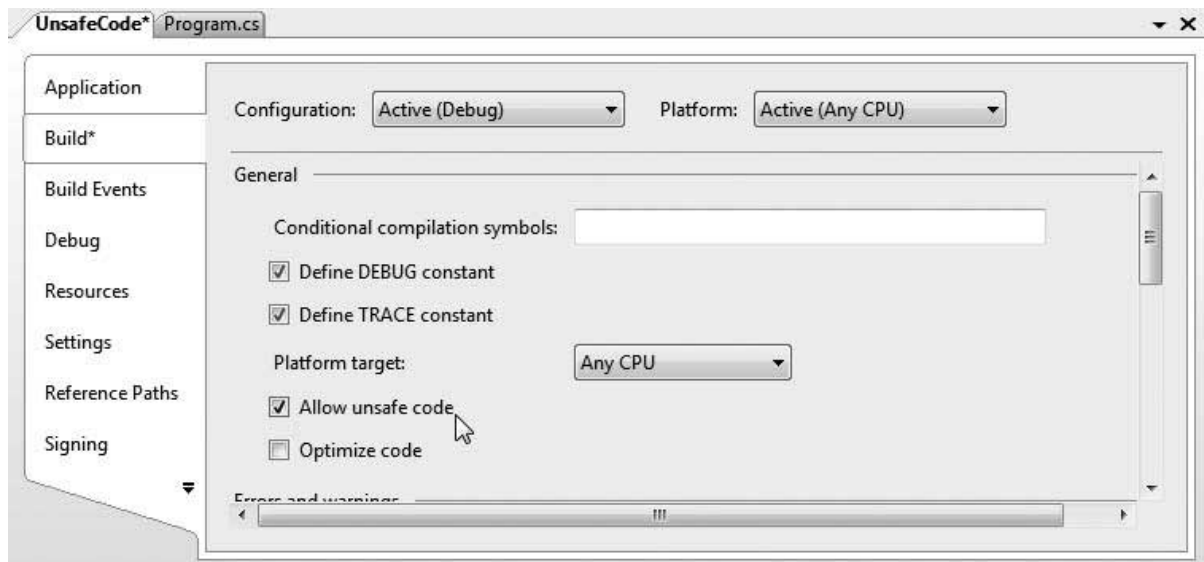
```
Public Overloads Overrides Function Equals(ByVal o As Object) As Boolean
    Return o.ToString() = Me.ToString()
End Function
Public Shared Operator =(ByVal p1 As Point, ByVal p2 As Point) As Boolean
    Return p1.Equals(p2)
End Operator
Public Shared Operator <>(ByVal p1 As Point, ByVal p2 As Point) As Boolean
    Return Not p1.Equals(p2)
End Operator
```

وهكذا تجد ان بامكانك عمل overloading لأي من هذه المعاملات:

```
+, -, !, ~, ++, --, true, false
+, -, *, /, %, &, |, ^, <<, >>
==, !=, <, >, <=, >=
```

13. المؤشرات Pointers

إذا كنت مبرمج C++ ، فأنت بالتأكيد تدرك معنى pointer، أما لو لم تكن كذلك، أما لو لم تكن كذلك فعليك ان تعرف انه بالاضافة إلى ال value type وال reference type، هناك ايضاً ال pointer type والتي تختص بالاشارة إلى اماكن معينة في الذاكرة.

بداية أول ما ستعرفه انك لن تتعامل معها في الكود الطبيعي ، بل ستضطر لاستخدام unsafe code لعمل ذلك حيث أن السي شارب لن تسمح لك افتراضياً بالتعامل مع ال pointers، لذا قم اولاً بالسماح لنفسك باستخدامها بالشكل التالي:

الصورة 12. 2. تمكين استخدام خاصية ال Unsafe code في المشروع.

والآن من خلال الكود الخاص بك يمكنك كتابة كود يتعامل مع ال pointers بالشكل التالي مثلاً:

كود	C#

```
class Program
{
    static void Main(string[] args)
    {
        unsafe
        {
            // Work with pointer types here!
        }
        // Can't work with pointers here!
    }
}
```

الآن سنتعرف على الأدوات الأساسية التي ستعيننا على التعامل مع ال pointers قبل ان نغوص في الأمثلة:

الأدوات	الاستخدام
*	تستخدم لتعريف pointer
&	لمعرفة عنوان المتغير في الذاكرة
->	تستخدم للوصول إلى حقل ما داخل الفئة التي يشير إليها ال pointer
++، --، +، -، ==، !=	التحرك ضمن المؤشرات والمقارنة وخلافه.

الجدول 12. 2. معاملات التعامل مع المؤشرات في السي شارب.

مع استمرارنا في هذا الدرس سوف تكتشف ان التعامل مع ال unsafe code لن يكون بنفس سهولة العمل في safe code من خلال سي شارب العادية .

و لكن لماذا استخدم ال unsafe code ؟

- أغراض تعليمية.
- استخدامك لبعض ال dll's او ال COM Components التي تعمل اصلاً من خلال pointers.
- محاولة تحسين اداء وسرعة مهمة معينة من خلال الوصول المباشر للذاكرة .

في مشروع التخرج الخاص بنا وأثناء قراءة pixels صورة ما لعمل بعض عمليات Image Processing كان بامكاننا استخدام دوال GetPixel و SetPixel الخاصة بالصور للقراءة

والكتابة ، ولكنها كانت لتكون عملية طويلة جداً للمرور على الـpixels بهذه الطريقة ، الحل البديل كان باستخدام unsafe code والتعامل مع ال pointers مباشرة للمرور على ال pixels الخاصة بالصورة ، جرب مثلاً كود لطرح صورتين بالطريقة GetPixel و SetPixel ، وبعد هذه الدرس جربها مرة أخرى باستخدام pointers، وأخبرني بالفارق ...

الآن سنعود مرة أخرى لنشرح من البداية...

يمكنك الآن تعريف pointer بالشكل التالي:

كود	C#

```
public Node* Left;
```

ليس هذا فقط ، بل بامكانك تعريف structure او class من النوع unsafe بالشكل التالي:

كود	C#

```
public unsafe struct Node
{
    public int Value;
    public Node* Left;
    public Node* Right;
}
```

أو دالة ايضاً:

كود	C#

```
unsafe static void SquareIntPointer(int* myIntPointer)
{
    // Square the value just for a test.
    *myIntPointer *= *myIntPointer;
}
```

ولما كنا قد استخدمنا * لتعريف المتغير في البارميتر ، إذن سنرسل البارميتر باستخدام & بالشكل التالي:

كود	C#

```
SquareIntPointer(&myInt2);
```

ايضاً للوصول إلى المتغير Value في ال Node سنستخدم -> بالشكل التالي مثلاً:

كود	C#

```
n->Value=5;
```

وماذا عن VB.net ؟

لو كنت مبرمج VB.net فلن يمكنك الاستفادة من المؤشرات مباشرة ، ولكن توفر لك الفئة Marshal التابعة لمجال الأسماء System.Runtime.InteropServices محاكاة قريبة من عالم المؤشرات ، لنفترض الكود التالي C# :

كود C#

```
Bitmap bitmap = new Bitmap(this.BackgroundImage);
unsafe
{

    System.Drawing.Imaging.BitmapData bmpData = bitmap.LockBits(new
Rectangle(0, 0, bitmap.Width, bitmap.Height),
                                        ImageLockMode.ReadWrite,
                                        PixelFormat.Format32bppArgb);
    byte * pixel = (byte*)(void*)bmpData.Scan0;
    //the last syntax is equivalent to:
    //byte* pixel = (byte*)bmpData.Scan0.ToPointer();
    pixel[0] = 255;
    pixel += 4;
    pixel[0] = 0;
}
```

سيتم تطبيقها في VB بالشكل التالي :

كود VB.NET

```
Dim bitmap As New Bitmap(Me.BackgroundImage)
Dim width As Integer = bitmap.Width
Dim height As Integer = bitmap.Height
Dim bmpData As System.Drawing.Imaging.BitmapData = _
                          bitmap.LockBits(New Rectangle(0, 0, width, height) _
                          , System.Drawing.Imaging.ImageLockMode.ReadWrite, _
                          PixelFormat.Format32bppArgb)
System.Runtime.InteropServices.Marshal.WriteByte(bmpData.Scan0,
bmpData.Stride, 255)
System.Runtime.InteropServices.Marshal.WriteByte(bmpData.Scan0,
bmpData.Stride + 4, 0)
bitmap.UnlockBits(bmpData)
```

*** مع الشكر للأخ وليد صاحب المثال .

14. Query Expressions

تعتبر ال Query Expressions هي الخطوة الأولى والأساسية في عالم LINQ ، سنتعرف عليها هنا باختصار شديد كميزة جديدة من مميزات net 2008. فيما سنؤجل باقي التفاصيل للفصول القادمة حينما نبدأ التعامل الفعلي مع قواعد البيانات.

لنرى هذا المثال مثلاً:

كود C#

```
from d in developers
where d.Language =="C#"
select d.Name;
```

كود VB.NET

```
From d In developers() _
Where(d.Language = "C#") _
Select d.Name
```

هذا بالضبط هو محتوى جملة الاستعلام التي تعودت على كتابتها سابقاً بالشكل التالي في ال SQL:

كود SQL

```
Select name from developers where language = "C#"
```

إذن لماذا هذا الشكل الجديد ؟

في السابق كنا نقوم بارسال جملة الاستعلام ليتم تنفيذها في قاعدة البيانات وتعود بناتج على شكل DataReader أو مهما يكن ، أما الآن اصبحت جمل الاستعلام جزء من محتويات اللغة التي تقوم ببرمجتها.

لتعمل على LINQ فلا بد من توريد مجال الأسماء هذا:

كود C#

```
using System.Linq;
```

VB.NET | كود

```
Imports System.Linq
```

تعتمد ال LINQ على ان قاعدة البيانات هي عبارة عن Array أو أي جزء منها ، لذا سنجرب بعض العمليات على ال Array ، لنفترض الشكل التالي مثلاً:

C# | كود

```
string[] userNames = {"Ahmed", "Ali","Mohammed", "Ahmed","Ramy", "Khaled"};
```

VB.NET | كود

```
Dim userNames As String() = {"Ahmed", "Ali", "Mohammed", "Ahmed", "Ramy",
"Khaled"}
```

*** لن اذكرك كثيراً بأنني هنا لا اهتم بـ LINQ قدر ما اهتم بال Query Expressions ، الفارق هو ان الأولى خاصة بالتعامل مع قواعد البيانات بمختلف انواعها اما الثانية فهي تعلمك كيفية الكتابة فقط دون التطرق لخصائص قواعد البيانات

هذه المصفوفة قد تكون قاعدة بيانات ، قد تكون ملف XML ، قد تكون أي شيء آخر ، الآن سنحاول بناء جملة استعلام لقراءة الاسماء التي تساوي"Ahmed"

C# | كود

```
IEnumerable<string> subset = from users in userNames
where users == "Ahmed" orderby users select users;
```

VB.NET | كود

```
Dim subset As IEnumerable = From users In userNames Where users =
"Ahmed"OrderBy users Select users
```

الناتج سيكون ايضاً array ، لذا يمكنك الآن طباعتها ببساطة بالشكل التالي:

C# | كود

```
foreach (string s in subset)
    Console.WriteLine("Item: {0}", s);
```

VB.NET | كود

```
For Each s As String In subset
    Console.WriteLine("Item: {0}", s)
Next
```

يمكنك بالطبع استخدام اي دالة من دوال C# ، لذا فالجملة التالية صحيحة لعرض الاسماء اطول من ثلاث حروف:

كود C#
```
IEnumerable<string> subset = from users in userNames
                             where users.Length > 3
                             orderby users
                             select users;
```

كود VB.NET
```
Dim subset As IEnumerable = From users In userNames Where (users.Length > 3)
OrderBy users Select users
```

ايضاً قد لا تكون قادراً على معرفة نوع البيانات الظاهر ، خصوصاً لو كنت تستعلم عن اكثر من حقل حيث ان الناتج لا بد ان يكون فئة ، هنا تظهر لنا فائدة ال var أو ال Dim بدون Data Type التي شرحناها سابقاً ، ايضاً ربما لا تعود جملة الاستعلام بنتيجة لذا سنسترجع هنا فائدة ال Nullable Types والتي شرحناها ايضاً في درس سابق.

طريقة أخرى لكتابة هذه الجمل باستخدام Lambda، وسنتعرف عليها في مرحلة قادمة.

15. Preprocessor Directives

كثيراً خلال تصفحك للبرامج او للمشاريع الجاهزة أو حتى للادوات في برنامجك ما تعثر على الشكل التالي مثلاً:

```
"Image Loader"

#region "Constructors"
    // a lot of code goes here.
#endregion
```

وكنت على حد علمك تعرف انها طريقة لوضع مجموعة من الاكواد ضمن حدود معينة بحيث يتم فتحها واغلاقها بسهولة لضمان عدم التشويش لك أثناء كتابتك الكود ، إلا ان ما ستعرفه في

هذا الدرس أن هذه ال Regions وخلافها هي مجموعة من ال Preprocessor Directives التي سنتعرف عليها تفصيلاً في هذا الدرس.

15. 1. #region, #endregion

تستخدم عادة لتنسيق مظهر الكود في ملف الشفرة الخاص بك كما ذكرنا سابقاً، يمكن كتابتها بالشكل التالي مثلاً:

كود C#

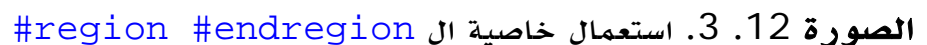
```
#region "Class Employee"

    public class Employee
    {
        private string _name;
        private string _age;
        private string _salary;
        public string name
        {
            get { return _name; }
            set { _name = value; }
        }
        public string age
        {
            get { return _age; }
            set { _age = value; }
        }
        public string name
        {
            get { return _salary; }
            set { _salary = value; }
        }
    }

#endregion
```

VB.NET	كود

```
#Region "Class Employee"

Public Class Employee

    Private _name As String
    Private _age As String
    Private _salary As String

    Public Property name() As String
        Get
            Return _name
        End Get
        Set(ByVal value As String)
            _name = value
        End Set
    End Property

    Public Property age() As String
        Get
            Return _age
        End Get
        Set(ByVal value As String)
            _age = value
        End Set
    End Property

    Public Property name() As String
        Get
            Return _salary
        End Get
        Set(ByVal value As String)
            _salary = value
        End Set
    End Property

End Class

#End Region
```

الآن يمكنك فتحها واغلاقها من الطرف ، بحيث يكون كودك منظماً بالشكل التالي مثلاً:

الصورة 12 .3. استعمال خاصية ال #region #endregion

15 .2. #if, #elif, #else, #endif

يطلق عليها اسم Conditional Code Compilation ، وتستخدم لتنفيذ اجزاء معينة من الكود في حالات معينة فقط ، فمثلاً لجعل جزء من الكود لا يعمل فقط إلا وقت ال Debug وليس في وقت ال Release نكتب الكود بالشكل التالي

كود	C#

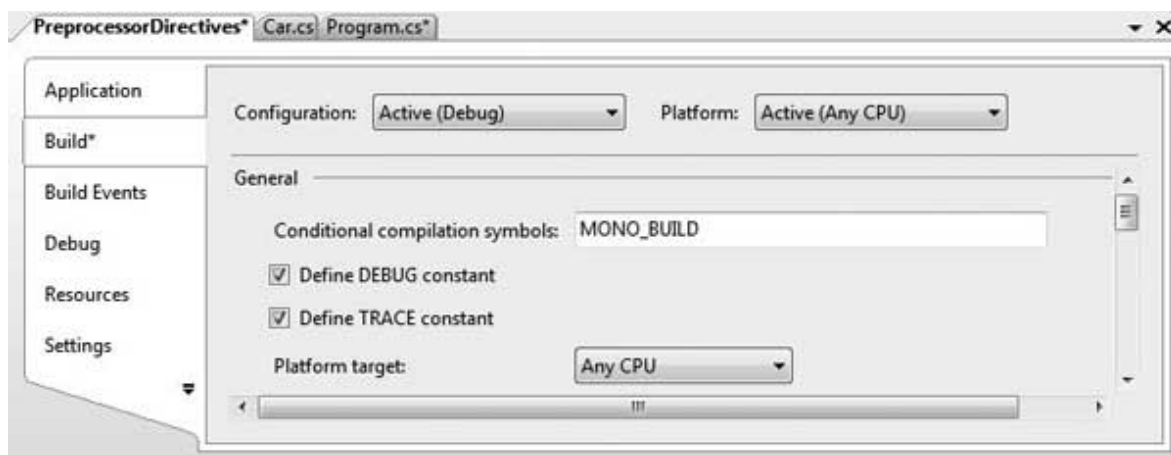
```
#if DEBUG
    Console.WriteLine("App directory: {0}", Environment.CurrentDirectory);
    Console.WriteLine("Box: {0}", Environment.MachineName);
    Console.WriteLine("OS: {0}", Environment.OSVersion);
    Console.WriteLine(".NET Version: {0}", Environment.Version);
#endif
```

كود	VB.NET

```
#If DEBUG Then
        Console.WriteLine("App directory: {0}", Environment.CurrentDirectory)
        Console.WriteLine("Box: {0}", Environment.MachineName)
        Console.WriteLine("OS: {0}", Environment.OSVersion)
        Console.WriteLine(".NET Version: {0}", Environment.Version)
#End If
```

بنفس الطريقة يمكن استخدام else و elif.

15. 3. #define, #undef

تستخدم لتعريف symbol معين ، مثلاً يمكنك تعريف symbol لل Debug ، او يمكنك تعريف symbol خاص بك بأي اسم ليتم استخدامه لاحقاً ، المثال التالي مثلا لتعريف Symbol يعني ان هذا الكود يتم عمل Debug له فقط على Mono .

الصورة 12. 4. خصائص المشروع، اضافة رموز ترجمة إلى المشروع.

لاحقاً يمكنك كتابة كود بالشكل التالي:

C# كود

```
#define DEBUG
#define MONO_BUILD

using System;

namespace PreprocessorDirectives
{
    class Program
    {
        static void Main(string[] args)
        {

#if MONO_BUILD
            Console.WriteLine("Compiling under Mono!");
#else
            Console.WriteLine("Compiling under Microsoft .NET");
#endif

        }
    }
}
```

VB.NET كود

```
#Define DEBUG
#Define MONO_BUILD
Imports System
Namespace PreprocessorDirectives
    Class Program
        Private Shared Sub Main(ByVal args As String())
#If MONO_BUILD Then
            Console.WriteLine("Compiling under Mono!")
#Else
            Console.WriteLine("Compiling under Microsoft .NET")
            Dim INDEXERS As n, OPERATORS As n, [AND] As n
            POINTERS()
#End If
        End Sub
    End Class
End Namespace
```

16. XML Commenting

في الواقع فإن عمل Comments للأكواد يعد أمراً في غاية الأهمية خصوصاً في حالة المشاريع الكبيرة ، حيث تستطيع مراقبة كودك كما يستطيع اي شخص آخر بقليل من الجهد معرفة كودك والاكمال عليه حتى ولو بعد توقفك عن العمل في نفس الكود بفترة طويلة جداً.

وكما عرفنا في بدايات هذه الدروس ، يتم عمل ال Comment بالشكل التالي

كود C#
```
// here we will do something, set x=startvalue
x = FirstClass.Default();
```

كود VB.NET
```
' here we will do something, set x=startvalue
x = FirstClass.Default()
```

الآن سنتعرف على طريقة جديدة ، تمكننا من كتابة ال Comments بأسلوب XML بما يوفر لنا عدة مزايا سنتعرف عليها خلال الدرس.

ملاحظة

من اوائل اللغات التي طبقت هذه الطريقة كانت Java من خلال javadoc.

يتم ذلك بداية من خلال وضع /// ، بعد وضع هذه العلامة تستطيع وضع أي علامات خاصة بك للكود والذي سيتم التعامل مع لاحقاً على أن XML ما دام يطبق مبادئ XML، هناك مجموعة من العناصر القياسية التي يفضل استخدامها لتوحيد المفاهيم.

العنصر	الاستخدام
<c>	لتحديد ان اللاحق لا بد أن يعرض بخط مختلف
<code>	لتحديد ان تعدد الاسطر سيتم التعامل معه ككود
<example>	تحديد مثال لشرح الكود المكتوب
<exception>	الملف الذي يحتوي على الاستثناءات والأخطاء التي يمكن ان تنتج عن هذا

<list>	ادراج قائمة جداول داخل الDocumentation
<param>	لشرح بارميتر معين
<permission>	وصف وسائل الحماية لجزء معين
<remarks>	خيارات الBuild
<returns>	وصف ناتج الدالة ;return
<see>	رابط آخر لجزء آخر من الDocumentation
<seealso>	مثل السابق، ولكن (انظر ايضاً)
<summary>	وصف اجمالي للجزء المشروح
<value>	لوصف خاصية معينة

الجدول 12. 3. وسوم تعليقات ال XML Documentation

والآن، ما هي الفائدة التي ستجنيها إذا استخدمت هذه الطريقة بدلاً من الطريقة التقليدية ؟؟؟

أولاً ، جرب مثلاً كتابة ما يلي لفئة الموظفين مثلاً:

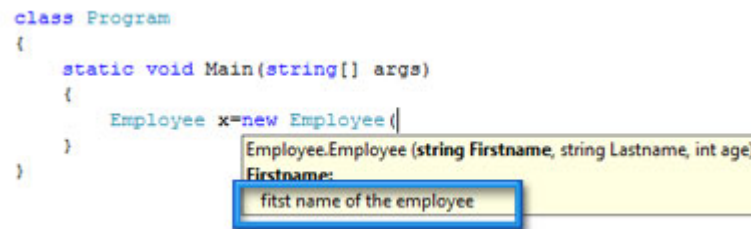
كود C#

```
/// <summary>
/// Employee Class of the company
/// </summary>
    partial class Employee
    {
        /// <summary>
        ///
        /// </summary>
        /// <param name="Firstname">first name of the employee</param>
        /// <param name="Lastname">last name of the employee</param>
        /// <param name="age">age of the employee</param>
        public Employee(string Firstname, string Lastname, int age)
        {
        }
}
```

كود	VB.NET

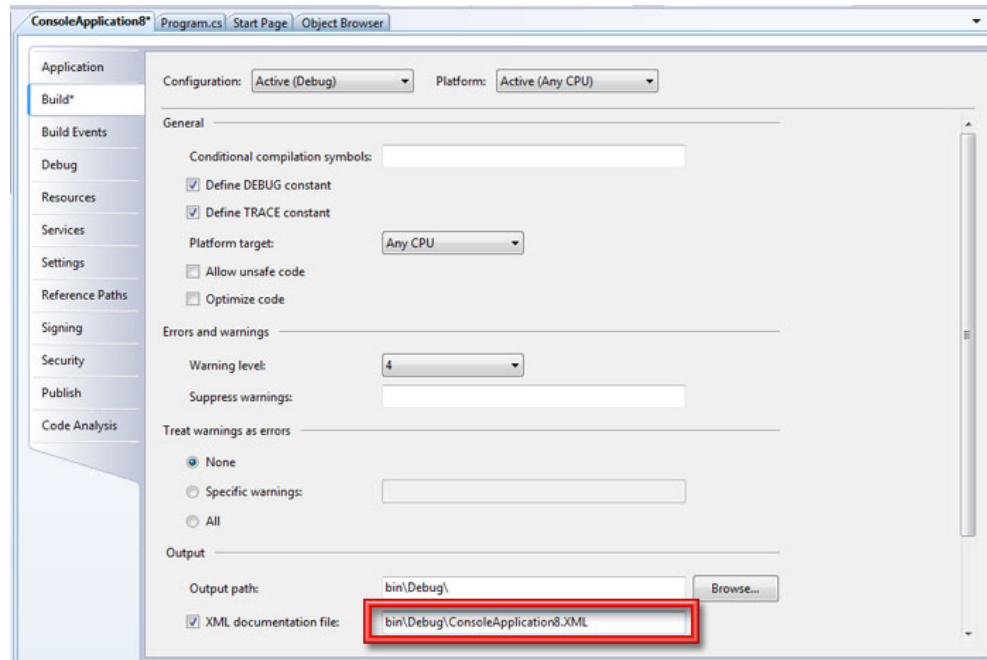
```
''' <summary>
''' Employee Class of the company
''' </summary>
Partial Class Employee
    ''' <summary>
    '''
    ''' </summary>
    ''' <param name="Firstname">first name of the employee</param>
    ''' <param name="Lastname">last name of the employee</param>
    ''' <param name="age">age of the employee</param>

    Public Sub New(ByVal Firstname As String, ByVal Lastname As String, ByVal
age As Integer)
    End Sub
End Class
```

وجرب الآن عمل الكود ، لاحظ الصورة التالية:

```
class Program
{
    static void Main(string[] args)
    {
        Employee x=new Employee (
    }
}
Employee.Employee (string Firstname, string Lastname, int age)
Firstname:
fitst name of the employee
```

الصورة 12. 5. استعمال تعليقات ال XML.

هل لاحظت الفارق ، اصبح الكود يظهر واضحاً لباقي مبرمجي فريقك ، اليس كذلك ؟

نقطة أخرى Documentation

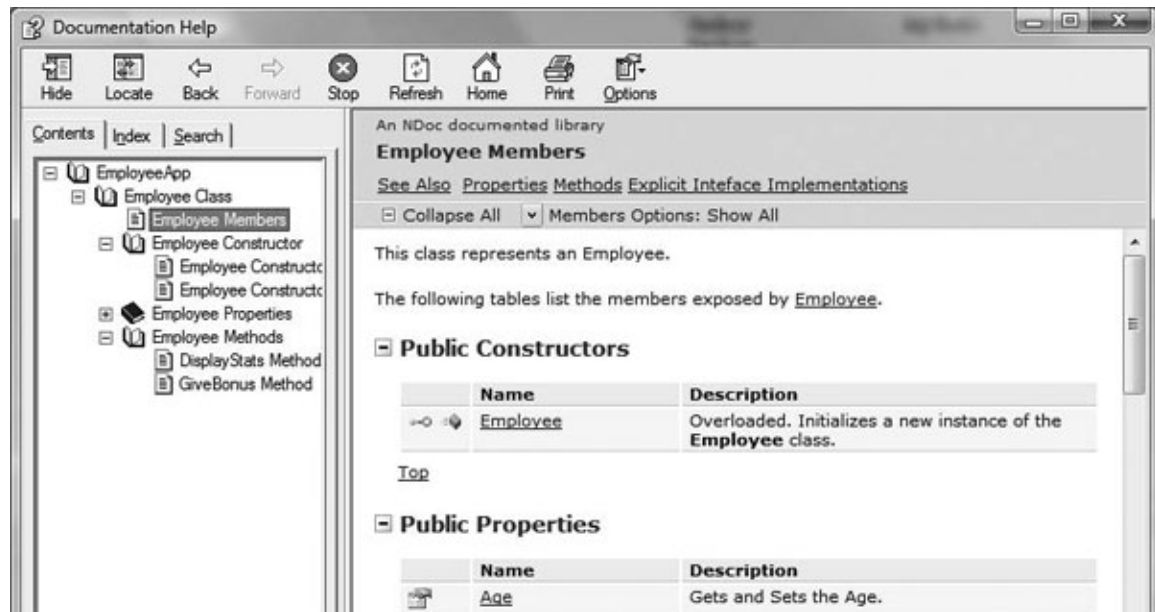
ايضاً ومن ضمن خيارات ال Build ، يمكنك اخراج Documentation كاملة لمشروعك اعتماداً على هذه الوسوم ، لذا من خصائص المشروع قم باختيار Build وقم باختيار انتاج ملف XML بالشكل التالي:

الصورة 12. 6. ضبط خيارات المشروع لبناء ملف ال Documentation

جرب الوصول لهذا الملف الذي قمت باختياره، ستجد شيئاً مثل هذا:

```
- <members>
  - <member name="T:WindowsFormsApplication11.Properties.Resources">
      <summary>A strongly-typed resource class, for looking up localized strings, etc.</summary>
    </member>
  - <member name="P:WindowsFormsApplication11.Properties.Resources.ResourceManager">
      <summary>Returns the cached ResourceManager instance used by this class.</summary>
    </member>
  - <member name="P:WindowsFormsApplication11.Properties.Resources.Culture">
      <summary>Overrides the current thread's CurrentUICulture property for all resource lookups using this strongly typed resource class.</summary>
    </member>
  - <member name="T:WindowsFormsApplication11.Employee">
      <summary>Employee Class of the comapny</summary>
    </member>
  - <member name="M:WindowsFormsApplication11.Employee.#ctor(System.String,System.String,System.Int32)">
      <summary />
      <param name="Firstname">fitst name of the employee</param>
      <param name="Lastname">last name of the employee</param>
      <param name="age">aqge of the employee</param>
    </member>
  - <member name="F:WindowsFormsApplication11.Form1.components">
      <summary>Required designer variable.</summary>
    </member>
  - <member name="M:WindowsFormsApplication11.Form1.Dispose(System.Boolean)">
      <summary>Clean up any resources being used.</summary>
      <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>
    </member>
  - <member name="M:WindowsFormsApplication11.Form1.InitializeComponent">
      <summary>Required method for Designer support - do not modify the contents of this method with the code editor.</summary>
    </member>
  - <member name="M:WindowsFormsApplication11.Program.Main">
      <summary>The main entry point for the application.</summary>
    </member>
  </members>
</doc>
```

الصورة 12. 7. ملف ال Documentation الناتج.

نقطة ثالثة Documentation مرة أخرى

للاسف لا يوفر Visual Studio 2008 اداة افتراضية لتحويل ال XML السابق لصيغة مفهومة ، لكن يمكن استخدام أداة مثل NDoc لتحويل الملف السابق إلى مثل هذه الصورة:

الصورة 12. 8. برنامج ال NDoc لتحويل ملفات ال Documetation إلى ملفات مساعدة (Help).

كل هذا فقط من ال Comments ...

يمكنك الوصول إلى نسخة على ال Sourceforge من الرابط التالي :

رابط

http://ndoc.sourceforge.net/

17. net Assemblies.

موعدنا هذه الفترة مع ال net assemblies. ، لن نطيل فيها كثيراً ولكننا سنعرف النقاط الاساسية فيها فيما يمكنك الاطلاع على المزيد عنها من خلال MSDN

17. 1. ال namespace

في نفس ال namespace يمكن لجميع المكونات تحته ان ترى بعضها البعض ، لذا دوماً وفي مشاريعك الجديدة اجعل namespace موحد لجميع مكونات برنامجك

كود	C#

```
namespace example
{
    class someclass
    {
        void method()
        {
        }
    }
}
```

كود	VB.NET

```
Namespace example
    Class someclass
        Private Sub method()

        End Sub
    End Class
End Namespace
```

أما لو كان لدينا فئة classCar في namespace باسم آخر، فلن يمكنك كتابة الكود التالي ضمن الدالة السابقة method

كود	C#

```
void method()
{
    classCar x = new classCar();
}
```

كود	VB.NET

```
Private Sub method()
    Dim x As New classCar()
End Sub
```

لكن يمكننا عمل import لل example لتتمكن من تعريف الفئة بالشكل التالي

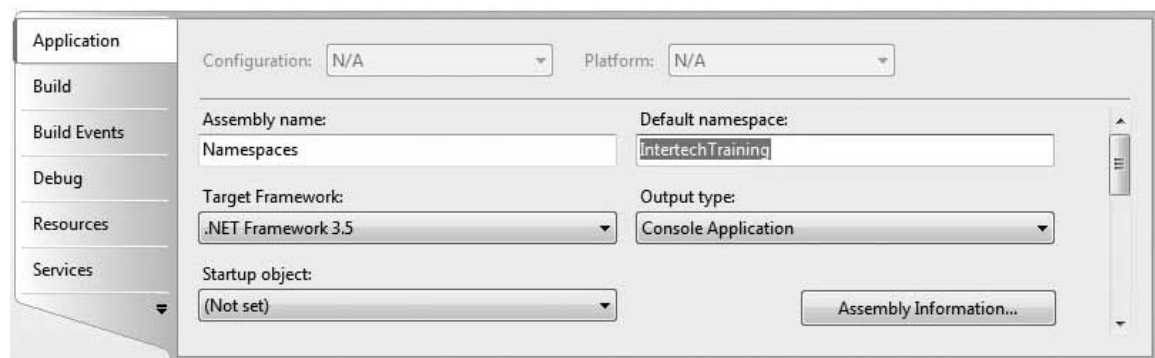
كود	C#

```
using example;
namespace example
{
    class someclass
    {
        void method()
        {
        }
        classCar x=new classCar();
    }
}
```

كود	VB.NET

```
Imports example
Namespace example
    Class someclass
        Private Sub method()

        End Sub
        Private x As New classCar()
    End Class
End Namespace
```

17. 2. تغيير ال Default Namespace

لتغيير مجال الاسماء الافتراضي:

الصورة 12. 9. تغيير مجال الأسماء الافتراضي من خصائص المشروع.

17 .3. شكل ملف الاسمبلي Format of a .NET Assembly

إذا قمت بفتح ملف الاسمبلي ب dumpbin.exe مثلاً يمكنك ان تلاحظ أن مكلف الاسمبلي يتكون من العناصر التالية:

Win32 file header

هنا ستجد معلومات عن نوع هذا الملف ، هل هو console ام GUI ام Dll وكيفية تنفيذه على نظم التشغيل من ويندوز .

CLR file header

يحتوي على المعلومات التي لا بد لأي net application. ان يدعمها ، يحتوي على طريقة كتابة ال managed file ووجود ال resources من عدمه وخلافه.

لا تهمك كثيراً المعلومات السابقة للنقطتين السابقتين ، لإنك لن تتعامل معها سوى إذا ما كنت تحتاج لبناء كومبايلر خاص بك تحت بيئة عمل net. .

CIL code

الكود الخاص ببرنامجك ، يتم ترجمته فوراً باستخدام JIT التي تحدثنا عنها سابقاً ، سيكون عنها درس مستقل إن شاء الله.

Type metadata

تحتوي على تفاصيل الانواع الخارجية التي تستخدمها في برنامجك.

An assembly manifest

الرابط بين ملفات الاسمبلي المختلفة ، تحدد اصدار الاسمبلي وخلافه.

Optional embedded resources

اي ملف اسمبلي يمكن ان يحتوي على اي عدد من ملفات الريسورس تشير إلى صور وايقونات وخلافه.

17. 4. Private Assemblies

ال Private Assemblies ملف اسمبلي يعمل من خلال مسار البرنامج او التطبيق ، حيث لن يبحث ال Visual Studio عن هذا الملف سواء في الريجستري او في أي مكان آخر ، فقط سيبحث في مسار البرنامج عنه.

في حالة حذف برنامجك يتم حذف هذه الملفات معه ، ايضاً يمكنك نقل التطبيق بعد عمل setup له مباشرة كونه لا يتعامل مع أي شيء يخص النظام.

ملفات الاسمبلي الافتراضية او ال dll التي تقوم بعملها هي من هذا النوع.

17. 5. Shared Assemblies

هذا هو النوع الثاني ، هنا لن تكون وحدك من يستخدم ملف ال dll هذا ، بل إن بإمكان الملف التعامل مع اكثر من تطبيق على نفس الجهاز ، مثلاً System.Windows.Forms.dl ، عادة ما تجدها في ملف الاسمبلي الموجود في الويندوز ولا يتم حذفها مع حذف برنامجك.

اول نقطة ستتعامل معها لإنشاء ملف اسمبلي من هذا النوع هو ضرورة وجود اسم unique لهذا الملف حتى لا يتضارب مع باقي الملفات الأخرى ، أيام الCOM كان هناك ما يعرف باسم COM globally unique identifier (GUID)، حيث يتم اعطاء اسماء مختلفة لكل COM جديد ، الآن اصبح عليك اعطاءه ما يعرف باسم strong name والذي لا يعدو كونه 128 بت من الارقام تشترك المكونات التالية في تحديده:

- اسم ملف الاسمبلي.

- نسخة ملف الاسمبلي.
- public key value -الموجودة في AssemblyKeyFile
- في حالة وجود اي اعدادت اقليمية في AssemblyCulture .
- Digital Signature يتم تكوينه باستخدام hash بين محتويات ملف الاسمبلي وال public key.

لعمل public key نستخدم البرنامج SDK's sn.exe بالشكل التالي مثلاً:

كود	Command Prompt
	sn -k MyTestKeyPair.snk

يمكنك ايضاً عمل ذلك مباشرة من خلال فيجوال ستوديو.نت من خلال Properties page ثم Signing.

18. المسارات المتعددة MultiThreading

18. 1. مقدمة

تعرف نظم التشغيل الحديثة بأنها multitasking systems وهو ما يعني امكانية تنفيذ اكثر من مهمة في نفس الوقت ، لذا تجد أن بامكانك تشغيل عدة برامج في نفس الوقت.

برامج net. من هذا النوع افتراضياً ، لكنك تلاحظ في بعض البرامج ان البرنامج الواحد قادر على تنفيذ اكثر من عملية في نفس الوقت دون أن يكون لهما تأثير متعارض ، هذا ما يعرف باسم multithreading.

لذا تجد ان برنامج الماسنجر يتيح لك ارسال ملفات والحديث واستخدام الكاميرا والتحدث مع اكثر من شخص بنفس الوقت ، وهو ما لم يكن الماسنجر قادراً عليه لو لم يكن يفصل هذه المهام داخل البرنامج الواحد عن بعضها ، هذا هو مضمون درسنا الحالي.

ملاحظة

جميع الأوامر التي سنتعامل معها هنا تقع تحت نطاق الأسماء System.Threading، لذا قم باستيراده أولاً

العناصر الاساسية داخل مجال الاسماء هذا هي:

العنصر	الوصف
Thread	لتعريف Thread جديد والتعامل معه
ThreadPool	مجموعة من ال Threads يمكن لها التعامل فيما بينها
ThreadState	Enum يحتوي على عدة حالات لأيThread
ThreadStart	بدء التنفيذ في Thread
ThreadPriority	تحديد اولوية هذا ال Thread

الجدول 12. 5. الفئات الرئيسية في مجال الأسماء System.Threading

بالاضافة إلى:

1. Semaphore
2. Mutex
3. Monitor

وهي مجموعة من خوارزميات التزامن التي سنتعرف عليها في الجزء التالي

18. 2. خوارزميات التزامن Synchronization

وهي في الاساس الجوريزمات تستخدم لعمليات التزامن Synchronization بحيث لا يسمح لأكثر من Thread بالوصول إلى نفس المصادر في نفس الوقت ، لمزيد من التعرف على هذه الالجوريزمات يمكن البدء من هنا:

رابط

http://en.wikipedia.org/wiki/Semaphore_(programming)
http://en.wikipedia.org/wiki/Mutual_exclusion
http://en.wikipedia.org/wiki/Monitor_(synchronization)

18. 3. System.Threading.Thread

الفئة الاساسية في مجال الاسماء هذا ، تتيح لنا انشاء threads وتنفيذ مهامنا المختلفة عليها ، مكونات هذه الفئة الاساسية هي:

المكون	الوصف
Sleep()	توقف عمل ال thread لفترة من الوقت
IsAlive	قيمة توضح إذا كان ال thread ما زال يعمل ام لا
IsBackground	إذا كان ال thread يعمل في الخلفية background
Priority	الأولوية الحالية
ThreadState	حالة ال thread
Name	اسم الthread
Abort()	خروج
Join()	توقف عمل ال thread حتى حدوث الحدث في Join()
Resume()	استئناف العمل بعد ايقافه
Start()	بدء العمل للمرة الأولى
Suspend()	ايقاف العمل مؤقتاً

الجدول 12. 6. اهم خصائص و دوال الفئة Thread

سنقوم بداية بعمل تجربة سريعة للاستدلال على معنى ان يتم تنفيذ مهمتين في نفس الوقت ، نعود بعدها لاستئناف شرح المفاهيم الخاصة بالموضوع.

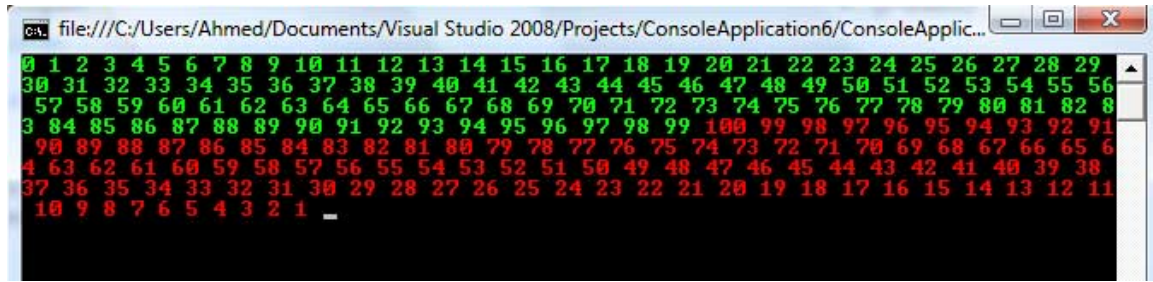
قم بتجربة الكود التالي:

كود	C#

```
static void Main(string[] args)
{
    order1();
    order2();

    Console.ReadKey();
}
static void order1()
{
    for (int i = 0; i < 100; i++)
    {
        Console.ForegroundColor = ConsoleColor.Green;
        Console.Write(i.ToString() + " ");
    }
}
static void order2()
{
    for (int i = 100; i > 0; i--)
    {
        Console.ForegroundColor = ConsoleColor.Red;
        Console.Write(i.ToString() + " ");
    }
}
```

كود	VB.NET

```
Private Shared Sub Main(ByVal args As String())
    order1()
    order2()
    Console.ReadKey()
End Sub
Private Shared Sub order1()
    For i As Integer = 0 To 99
        Console.ForegroundColor = ConsoleColor.Green
        Console.Write(i.ToString() + " ")
    Next
End Sub
Private Shared Sub order2()
    For i As Integer = 100 To 1 Step -1
        Console.ForegroundColor = ConsoleColor.Red
        Console.Write(i.ToString() + " ")
    Next
End Sub
```

الكود كما هو واضح يقوم بطباعة الأرقام تصاعدياً وتنازلياً ، الناتج الطبيعي هو طباعة التصاعدي ومن ثم التنازلي ، في حين يتم طباعة نتائج الدالة الأولى بالأخضر والثانية بالأحمر للتفريق لتكون شاشة النتائج بالشكل التالي مثلاً:

الصورة 12. 10. نتائج التنفيذ.

وهذا الطبيعي ، يتم تنفيذ الدالة الأولى حتى الانتهاء ومن ثم الثانية حتى الانتهاء ، أما الآن سنقوم بتعريف threads مختلفة للتنفيذ بحيث يتم تنفيذ كل دالة على واحد منها حيث سيتم تنفيذهم على البروسيسور في نفس الوقت ، الكود التالي مثلاً:

C#	كود

```
static void Main(string[] args)
{
    order1();
    order2();

    Console.ReadKey();
}
static void order1()
{
    for (int i = 0; i < 100; i++)
    {
        Console.ForegroundColor = ConsoleColor.Green;
        Console.Write(i.ToString() + " ");
    }
}
static void order2()
{
    for (int i = 100; i > 0; i--)
    {
        Console.ForegroundColor = ConsoleColor.Red;
        Console.Write(i.ToString() + " ");
    }
}
```

VB.NET	كود

```
Private Shared Sub Main(ByVal args As String())
    Dim t1 As New System.Threading.Thread(order1)
    t1.Start()
    Dim t2 As New System.Threading.Thread(order2)
    t2.Start()
    Console.ReadKey()
End Sub
Private Shared Sub order1()
    For i As Integer = 0 To 99
        Console.ForegroundColor = ConsoleColor.Green
        Console.Write(i.ToString() + " ")
    Next
End Sub
Private Shared Sub order2()
    For i As Integer = 100 To 1 Step -1
        Console.ForegroundColor = ConsoleColor.Red
        Console.Write(i.ToString() + " ")
    Next
End Sub
```

الآن لنرى طبيعة النتائج.

```
file:///C:/Users/Ahmed/Documents/Visual Studio 2008/Projects/ConsoleApplication6/ConsoleApplic...
0 1 100 2 99 3 98 97 96 4 95 5 94 6 93 7 92 8 91 9 90 10 89 11 88 12 87 13 86 14
 85 15 84 16 83 17 82 18 81 19 80 20 79 21 78 22 77 23 76 24 75 25 74 73 26 72 2
7 71 28 70 29 69 30 68 31 67 32 66 33 65 34 64 35 63 36 62 37 61 38 60 39 59 40
58 41 57 42 56 43 55 44 54 45 53 46 52 47 51 48 50 49 49 50 48 51 47 52 46 53 45
 54 44 55 43 56 42 57 41 58 40 59 39 60 38 61 37 62 36 63 35 64 65 66 34 67 33 6
8 32 69 31 70 30 71 29 72 28 73 27 74 26 75 25 76 24 77 23 78 22 79 21 80 20 81
19 82 18 83 17 84 16 85 15 86 14 87 13 88 12 89 11 90 10 91 9 92 8 93 7 94 6 95
5 96 4 97 3 98 2 99 1
```

الصورة 12. 11. نتائج التنفيذ.

بالطبع النتائج لن تكون كما هي كل مرة ، جرب تغيير ال Priority مثلاً لواحد منهم وجرب النتائج ، ستجد أن صاحب الأولوية الأعلى يتم الانتهاء منه قبل الثاني ، الكود التالي مثلاً:

C#	كود

```
t1.Priority = System.Threading.ThreadPriority.Highest;
```

VB.NET	كود

```
t1.Priority = System.Threading.ThreadPriority.Highest
```

ستصبح النتائج بالشكل التالي:

```
file:///C:/Users/Ahmed/Documents/Visual Studio 2008/Projects/ConsoleApplication6/ConsoleApplic...
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
30 31 32 33 34 35 36 37 38 39 40 41 42 43 100 99 98 44 97 45 96 46 95 47 94 48 9
3 49 92 50 91 51 90 52 89 53 88 54 87 55 86 56 85 57 84 58 83 59 82 60 81 61 62
63 64 65 66 67 68 69 70 71 72 73 74 75 76 80 79 77 78 77 76 78 75 79 74 80 73 81
 72 82 71 70 69 68 83 67 84 66 85 65 86 64 87 63 88 62 61 60 89 59 90 58 91 57 9
2 56 93 55 94 54 95 53 52 96 51 97 50 49 48 98 47 99 46 45 44 43 42 41 40 39 38
37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11
 10 9 8 7 6 5 4 3 2 1
```

الصورة 12. 12. نتائج التنفيذ.

ستجد اختلافات في التنفيذ ، لكن الشاهد هو أن الكودين تم تنفيذهم سوية في هذا الوقت.

ملاحظة

جميع الأوامر التي سنتعامل معها هنا تقع تحت مجال الأسماء System.Threading، **لذا قم باستيراده أولاً**

18. 4. الأولوية Priority

تحدد ال Priority أولوية التنفيذ عندما يتم ادخال ال threads على البروسيسور، حيث ان ال thread ذو الاولوية الأعلى يحظى بعدد مرات تنفيذ ، لتقريب المفهوم نفترض ان لدينا :

1. مهمة 1 : اولوية قصوى.
2. مهمة 2 : اولوية قصوى.

في هذه الحالة يتم ادخال المهمة 1 للبروسيسور، ثم 2 ، ثم 1 وهكذا.

أما في حال كون مهمة 2 ذات اولوية اقل ، يكون الأمر بالشكل التالي:

المهمة 1 ، المهمة 1 ، المهمة 1 ، المهمة 2 ، المهمة 1 ، المهمة 1 ، المهمة 1 ، المهمة 2 ...

وهكذا حتى الانتهاء من احدهما.

ليس هذا مكان شرح الجورزمات البروسيسور، إنما لو أردت الزيادة يمكنك البدء من هنا، حيث تجد عدة انواع من ال scheduling :

رابط

http://en.wikipedia.org/wiki/Scheduling_%28computing%29

نعود مرة أخرى ، لتحديد Priority أي مهمة لدينا نستخدم ال enum التالي:

كود C#

```
public enum ThreadPriority
{
    AboveNormal,
    BelowNormal,
    Highest,
    Idle,
    Lowest,
    Normal, // Default value.
    TimeCritical
}
```

كود VB.NET

```
Public Enum ThreadPriority
    AboveNormal
    BelowNormal
    Highest
    Idle
    Lowest
    Normal
    ' Default value.
    TimeCritical
End Enum
```

ويصبح الكود بالشكل التالي مثلاً:

كود C#

```
t1.Priority = System.Threading.ThreadPriority.Highest;
```

كود VB.NET

```
t1.Priority = System.Threading.ThreadPriority.Highest
```

قبل البعد عن المواضيع الاساسية ، لا تنسى ان بامكانك استخدام Sleep() لايقاف التنفيذ لمدة، suppose لايقاف مؤقت ... الخ من النقاط التي بدأنا بها شرحنا لهذا الدرس.

18 .5 ParameterizedThreadStart

لعلك لاحظت في المثال السابق اننا نمرر الدالة ومن ثم نقوم بعمل ()Start لها لتنفيذها في thread منفصل ، لكن ماذا لو كانت هذه الدالة تستقبل بارميترات ؟

الحل بسيط ، باستخدام ParameterizedThreadStart بالشكل التالي مثلاً:

كود — C#

```
Thread t = new Thread(new ParameterizedThreadStart(functionname));
t.Start(parms);
```

كود — VB.NET

```
Dim t As New Thread(New ParameterizedThreadStar(functionname))
t.Start(parms)
```

ملاحظة

لو كنت تود استخدام دالة تعود بقيمة فلا يمكن استخدام ThreadStart ولكن يمكن استخدام ()BeginInvoke وقراءة الناتج في ()EndInvoke

18 .6 Foreground and background

هناك نوعان من ال threads:

Foreground Thread: هذا يعني ان البرنامج لا يمكن ان يغلق حتى يتم الانتهاء من تنفيذ جميع ال Foreground threads الموجودة فيه ، النوع الافتراضي لأي thread تقوم بانشاءه هو من هذا النوع.

Background Thread: هذا يعني ان البرنامج يمكن ان يتم اغلاقه حتى لو لم يتم تنفيذ كافة ال background threads ، يتم عمله بالشكل التالي مثلاً:

كود — C#

```
t.IsBackground = true;
```

كود	VB.NET

```
t.IsBackground = True
```

18. 7. Threads Synchronization

عندما تعمل مع ال Threads فلا أحد يضمن ألا يحصل تداخل بين مسارات عملك المختلفة، لذا تم ايجاد ما يعرف بـ SyncLock والذي يطبق مبادئ التزامن التي اشرنا إليها مختصراً في درس سابق .

يتم استخدام SyncLock…End SyncLock والتي يتم تضمين الكود داخلها كي لا يحصل أي تداخل ، حيث تقوم باغلاق مصادر هذا المسار ولا تسمح لأي كود في نفس مجال الأسماء بالتداخل إلا بعد الإنتهاء ، بالشكل التالي مثلاً :

كود	C#

```
lock (myLock)
{
    for (i = 0; i <= 1000; i++)
    {
        Console.WriteLine(i.ToString());
    }
    Console.Write(number);
}
```

كود	VB.NET

```
SyncLock myLock
    For i = 0 To 1000
        Console.WriteLine(i.ToString())
    Next i
    Console.Write(number)
End SyncLock
```

حيث لن يسمح لأي مسار يستخدم المتغير myLock ان يعمل إلا بعد انتهاء حلقة التكرار وأمر الطباعة أيضاً ، لذا قم بوضعها في كل مكان تخشى أن يحصل فيه تداخل لقيمة المتغير number مثلاً .

يمكنك معرفة المزيد عن التزامن في المسارات من الدرس التالي – باللغة العربية - :

رابط

http://vb4arab.com/vb/showthread.php?t=6341

18 .8. ThreadPool

بحيرة المسارات – إن صح التعبير – هي البديل لكثرة المسارات في برنامجك وما تتسبب به من تشويش في البرنامج والكثير من الخلط بسبب أوامر التزامن والمصادر وخلافه ، يتم ذلك بتكوين مجموعة من المسارات المشتركة ، ويكون ذلك أكثر افادة في حالة المسارات التي لا يتم تنفيذها بكثرة في البرنامج :

كود C#

```
for (int i = 1; i <= 5; i++)
{
    ThreadPool.QueueUserWorkItem(sometask.Execute);
}
```

كود VB.NET

```
For i As Integer = 1 To 5
    ThreadPool.QueueUserWorkItem(AddressOf sometask.Execute)
Next
```

يمكنك معرفة المزيد عن هذا الموضوع من خلال الدرس التالي – باللغة العربية - :

رابط

http://vb4arab.com/vb/showthread.php?t=6340

18 .9. BackgroundWorker

تستخدم ال BackgroundWorker لتنفيذ مهمة معينة تأخذ وقتاً طويلاً بعيداً عن المسار الاساسي للبرنامج ، من أمثلة ذلك الدوال الخاصة بالقراءة من web service أو عمليات معالجة الصور او جلب بعض البيانات من كومبيوتر آخر أو تنفيذ عملية بحث ، او اجراء مجموعة من العمليات طويلة الأمد.

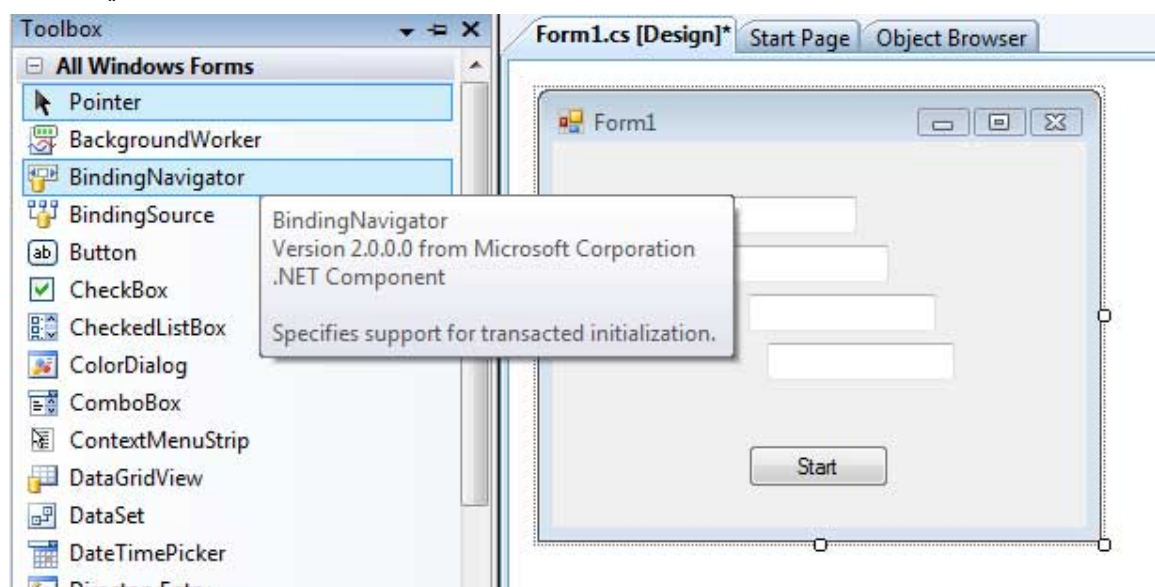
ومع أنك كان بإمكانك عمل هذه الدوال عن طريق تنفيذها في thread تقليدي ، إلا أن BackgroundWorker تعطيك مزيد من التحكم ، ببساطة يمكنك اخباره ال بالدالة التي ترغب في تنفيذها ومن ثم تشغيلها عن طريق RunWorkerAsync()، أخيراً بعد انتهاء التنفيذ يتم تنفيذ حدث RunWorkerCompleted والتي يمكنك فيه مثلاً عرض النتائج بعد انتهاء تنفيذ هذه المهمة.

كود C#

```
for (int i = 1; i <= 5; i++)
{
    ThreadPool.QueueUserWorkItem(sometask.Execute);
}
```

كود VB.NET

```
For i As Integer = 1 To 5
    ThreadPool.QueueUserWorkItem(AddressOf sometask.Execute)
Next
```

للبدء بالعمل قم بعمل Windows Form ، قم برسم بعض الادوات ومن ثم ضع زر امر للبدء بتنفيذ المهمة ، واخيراً قم بسحب أداة BackgroundWorker بالشكل التالي مثلاً:

الصورة 12. 12. الأداة BackgroundWorker

أهم دالتين هما هنا DoWork والذي يتم استدعاءه وقت بدء التنفيذ ، والحدث RunWorkerCompleted والذي يتم اطلاقه بعد الانتهاء من التنفيذ بالشكل التالي مثلاً:

كود	C#

```
private void ProcessNumbersBackgroundWorker_DoWork(object sender,
DoWorkEventArgs e)
{
}
private void ProcessNumbersBackgroundWorker_RunWorkerCompleted(object sender,
RunWorkerCompletedEventArgs e)
{
}
```

كود	VB.NET

```
Private Sub ProcessNumbersBackgroundWorker_DoWork(ByVal sender As Object, ByVal
e As DoWorkEventArgs)
End Sub
Private Sub ProcessNumbersBackgroundWorker_RunWorkerCompleted(ByVal sender As
Object, ByVal e As RunWorkerCompletedEventArgs)
End Sub
```

لاحقاً يتم البدء بالتنفيذ بالشكل التالي:

كود	C#

```
ProcessNumbersBackgroundWorker.RunWorkerAsync(args);
```

كود	VB.NET

```
ProcessNumbersBackgroundWorker.RunWorkerAsync(args)
```

حيث يتم تنفيذ الكود الموجود في الحدث DoWork .

الباب

13

الادخال و الاخراج في

.net System.IO

1. الفئات الأساسية في System.IO

الفئة	الوصف
BinaryReader BinaryWriter	تتيح لك قراءة وكتابة بياناتك على شكل Binary Files
BufferedStream	مخزن مؤقت للبيانات Buffer التي يمكن أن تأخذ طريقها لاحقاً إلى الذاكرة
Directory DirectoryInfo	مختص بالتعامل مع المجلدات والمعلومات المتعلقة بها
DriveInfo	مختص بالتعامل ومعرفة المعلومات عن وحدات التخزين في جهازك Driver
File FileInfo	مختص بالتعامل مع الملفات وكل ما يتعلق بها
FileStream	يمكنك من الوصول للملفات وعرض البيانات على شكل Stream
FileSystemWatcher	مراقبة ملف او مجلد واخبارك بالتعديلات التي تطرأ عليه
Path	مختص بالتعامل مع المتغيرات النصية لتحويل المسارات إلى صور مفهومة لنظام التشغيل
StreamReader StreamWriter	القراءة والكتابة إلى الملفات بصورة نصية
StringReader StringWriter	نفس السابق ، فقط مع اختلاف ان ال Reader وال Writer يتم انشاءه من الString

الجدول 13. 1. أهم فئات مجال الأسماء System.IO

هناك فئات اخرى في هذا المجال من الاسماء ، ولكن هذه هي أشهرها وأكثرها استخداماً.

2. الفوارق بين Directory و DirectoryInfo

الفارق الأساسي هو في ال structure الخاص بها ، حيث ان Directory مشتقة مباشرة منObject اما DirectoryInfo فهي مشتقة من FileStreamInfo المشتقة بدورها من ال Object ، اما الفارق في التعامل فهو ان الفئة Directory تعرف دوالها من النوع static بحيث يمكن استخدامها مباشرة ، اما مع DirectoryInfo فالامر مختلف حيث يتطلب الامر تعريف نسخة قبل استخدام الدوال.

نفس الامر ينطبق بالتبعية على الفرق بين File و... FileInfo

كلا الفئتين DirectoryInfo و FileInfo تحمل معها الخصائص الاساسية ل FileStreamInfo اضافة لخصائص التعامل مع الملفات والمجلدات ، وهي:

الخاصية	الوصف
Attributes	تختص بتمرير قيم اضافية للمجلد أو الملف عن طريق ال enumeration المسمى FileAttributes
CreationTime	قراءة وكتابة تاريخ الانشاء
Exists	معرفة هل الملف موجود أم لا
Extension	تعيد امتداد الملف
FullName	الحصول على المسار كاملاً للملف أو للمجلد
LastAccessTime	قراءة وكتابة آخر تاريخ للدخول على الملف او المجلد
LastWriteTime	قراءة وكتابة آخر تاريخ للكتابة في هذا الملف او المجلد
Name	معرفة اسم الملف أو المجلد

الجدول 13. 2. الخصائص المشتركة.

3. الفئة DirectoryInfo

اضافة لاحتواءها على الخصائص السابقة من FileStreamInfo، تحتوي هذه الفئة على الدوال التالية:

الدالة أو الخاصية	الوصف
Create()	انشاء مجلد او مجلد فرعي في المسار المحدد
CreateSubdirectory()	
Delete()	حذف المجلد وكامل محتوياته
GetDirectories()	تعيد مصفوفة من الاسماء توضح اسماء المجلدات الفرعية
GetFiles()	تعيد مصفوفة Array من FileInfo تحتوي على كافة
MoveTo()	نقل المجلد وكافة محتوياته إلى مكان جديد
Parent	معرفة المجلد الاكبر من هذا المجلد
Root	لمعرفة ال Driver الخاص بهذا المجلد

الجدول 13. 3. أهم خصائص و دوال الفئة.

قبل البدء ، انت بحاجة لتعريف نسخة من هذه الفئة ، يمكنك تمرير المكان الذي تود البدء منه في المشيد Constructor بالشكل التالي مثلاً (لمسار البرنامج) :

كود C#
```
DirectoryInfo dir1 = new DirectoryInfo(".");
```

كود VB.NET
```
Dim dir1 As New DirectoryInfo(".")
```

او إلى مسار عادي:

كود C#
```
DirectoryInfo dir1 = new DirectoryInfo("C:\Ahmed");
```

كود VB.NET
```
Dim dir1 As New DirectoryInfo("C:\Ahmed")
```

*** لو كنت مبرمج سي شارب فلا بد من عدم وضع \ وحدها داخل علامات التنصيص لإن لها استخدامات أخرى مثل \r \n وخلافه ، لكن يمكنك تحديد نوع البيانات بالداخل بأنها مباشرة عن طريق اضافة @ بالشكل التالي مثلاً:

كود C#
```
DirectoryInfo dir1 = new DirectoryInfo(@"C:\Ahmed");
```

ليس فقط بامكانك الربط مع مجلد موجود ، بل يمكنك الربط إلى مجلد غير موجود ومن ثم انشاءه:

كود C#
```
DirectoryInfo dir1 = new DirectoryInfo(@"C:\Ahmed\Test");
dir1.Create();
```

كود VB.NET
```
Dim dir1 As New DirectoryInfo("C:\Ahmed\Test")
dir1.Create()
```

يمكنك ايضاً استعراض كافة الملفات داخل المجلد:

كود C#
```
FileInfo[] imageFiles = dir.GetFiles();
```

كود VB.NET
```
Dim imageFiles As FileInfo() = dir.GetFiles()
```

يمكنك استخدام خصائص FileSystemInfo التي تحتوي على كافة المعلومات المتعلقة بالمجلد، فمثلاً لعرض تاريخ إنشاء المجلد:

كود C#
```
Console.WriteLine(dir1.CreationTime);
```

كود VB.NET
```
Console.WriteLine(dir1.CreationTime)
```

او يمكنك استعراض ملفات من نوع معين فقط ، مثلاً لاستعراض ملفات الصور فقط GIF ومن ثم كتابتها ، لا تنس ان الناتج من نوع FileInfo ويمكن تطبيق خصائص FileSystemInfo عليها.

كود C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;
namespace ConsoleApplication7
{
    class Program
    {
        static void Main(string[] args)
        {
            DirectoryInfo dir1 = new
DirectoryInfo(@"C:\Users\Ahmed\Documents\Visual Studio
2008\WebSites\WebSite4");
            FileInfo[] imageFiles = dir1.GetFiles("*.gif");
            foreach (FileInfo f in imageFiles)
            {
                Console.WriteLine("Name: {0} - Creation Date: {1} . \n\n",
f.Name, f.CreationTime);
            }
            Console.ReadKey();
        }
    }
}
```

كود VB.NET

```
Imports System
Imports System.Collections.Generic
Imports System.Linq
Imports System.Text
Imports System.IO
Namespace ConsoleApplication7
    Class Program
        Private Shared Sub Main(ByVal args As String())
            Dim dir1 As New DirectoryInfo("C:\Users\Ahmed\Documents\Visual
Studio 2008\WebSites\WebSite4")
            Dim imageFiles As FileInfo() = dir1.GetFiles("*.gif")

            For Each f As FileInfo In imageFiles
                Console.WriteLine("Name: {0} - Creation Date: {1} . " & Chr(10)
& "" & Chr(10) & "", f.Name, f.CreationTime)
            Next

            Console.ReadKey()
        End Sub
    End Class
End Namespace
```

الناتج سيكون بالشكل التالي مثلاً:

```
file:///C:/Users/Ahmed/Documents/Visual Studio 2008/Projects/ConsoleApplication7/ConsoleApplic...
Name: back.gif - Creation Date: 6/1/2008 5:56:28 PM .

Name: false.gif - Creation Date: 5/22/2008 2:16:12 PM .

Name: missed.gif - Creation Date: 5/22/2008 2:18:46 PM .

Name: score.gif - Creation Date: 5/22/2008 2:16:12 PM .

Name: Start.gif - Creation Date: 6/1/2008 5:47:51 PM .

Name: start2.gif - Creation Date: 6/1/2008 5:59:17 PM .

Name: true.gif - Creation Date: 5/22/2008 2:16:12 PM .
```

الصورة 13. 1. نتائج تنفيذ البرنامج على الشاشة

4. التعامل مع الفئة Directory

لا شيء جديد، فقط ستفقد الخصائص المتاحة من خلال الفئة FileSystemInfo وفي المقابل لن تكون محتاجاً لتعريف نسخة قبل البدء بالعمل ، لذا يمكنك العمل على الفئة مباشرة بالشكل التالي مثلاً لعملية الحذف

C#	كود

```
System.IO.Directory.Delete(@"C:\ahmed");
```

VB.NET	كود

```
System.IO.Directory.Delete("C:\ahmed")
```

5. التعامل مع الفئة DriveInfo

تتيح لك هذه الفئة استعراض ال Drivers في جهازك ومعرفة بعض المعلومات عنها ، هذا المثال مباشرة من كتاب Pro.CSharp 2008:

كود C#

```
Console.WriteLine("***** Fun with DriveInfo *****\n");
// Get info regarding all drives.
DriveInfo[] myDrives = DriveInfo.GetDrives();
// Now print drive stats.
foreach (DriveInfo d in myDrives)
{
    Console.WriteLine("Name: {0}", d.Name);
    Console.WriteLine("Type: {0}", d.DriveType);
    // Check to see whether the drive is mounted.
    if (d.IsReady)
    {
        Console.WriteLine("Free space: {0}", d.TotalFreeSpace);
        Console.WriteLine("Format: {0}", d.DriveFormat);
        Console.WriteLine("Label: {0}", d.VolumeLabel);
        Console.WriteLine();
    }
}
Console.ReadLine();
```

كود VB.NET

```
Console.WriteLine("***** Fun with DriveInfo *****" & Chr(10) & "")
' Get info regarding all drives.
Dim myDrives As DriveInfo() = DriveInfo.GetDrives()
' Now print drive stats.
For Each d As DriveInfo In myDrives
    Console.WriteLine("Name: {0}", d.Name)
    Console.WriteLine("Type: {0}", d.DriveType)
    ' Check to see whether the drive is mounted.
    If d.IsReady Then
        Console.WriteLine("Free space: {0}", d.TotalFreeSpace)
        Console.WriteLine("Format: {0}", d.DriveFormat)
        Console.WriteLine("Label: {0}", d.VolumeLabel)
        Console.WriteLine()
    End If
Next
Console.ReadLine()
```

6. التعامل مع الفئة FileInfo

الدوال الرئيسية في هذه الفئة - اضافة بالطبع للخصائص العادية - :

الخاصية او الدالة	الوصف
AppendText()	انشاء StreamWritter للكتابة في الملف
CopyTo()	نسخ الملف
Create()	انشاء ملف واعادة كائن FileStream
CreateText()	انشاء StreamWriter للكتابة في الملف
Delete()	حذف الملف
Directory	معرفة اسم المجلد
DirectoryName	معرفة مسار المجلد
Length	معرفة حجم الملف
MoveTo()	النقل ، يمكنك تحديد اسم جديد للملف المنقول
Name	اسم الملف
Open()	فتح الملف مع امكانية القراءة والكتابة وخلافه
OpenRead()	فتح الملف للقراءة فقط
OpenText()	فتح الملف باستخدام StreamReader
OpenWrite()	فتح الملف للكتابة فقط

الجدول 13. 4. أهم خصائص و دوال الفئة.

بعيداً عن خواص StreamReader والتي سنتعرف عليها تفصيلاً في جزء لاحق من هذه الدروس. سنتعرف على باقي الخصائص.

لانشاء ملف مثلاً باستخدام Create() :

كود	C#

```
FileInfo f = new FileInfo(@"C:\Test.txt");
FileStream fs = f.Create();
```

كود	VB.NET

```
Dim f As New FileInfo("C:\Test.txt")
Dim fs As FileStream = f.Create()
```

طبعاً لا تنس أن بامكانك استخدام أي من الخصائص الموجودة في FileSystemInfo كما سبق ، فقط الخاصية Attributes هي خاصية مميزة نوعاً ما حيث تشمل عدة تفاصيل ، حيث يمكنك من خلالها تحديد إذا كان الملف مخفي ام ظاهر ، للقراءة فقط ... الخ.

مثلاً لمعرفة هل الملف مخفي أم لا ؟

كود	C#

```
if ((File.GetAttributes(path) & FileAttributes.Hidden) ==
FileAttributes.Hidden)
{

}
```

كود	VB.NET

```
If (File.GetAttributes(Path) And FileAttributes.Hidden) = FileAttributes.Hidden
Then

End If
```

أما لاخفاء الملف نستخدم المعامل Or بالشكل التالي مثلاً:

كود	C#

```
File.SetAttributes(path, File.GetAttributes(path) | FileAttributes.Hidden);
```

كود	VB.NET

```
File.SetAttributes(path, File.GetAttributes(path) Or FileAttributes.Hidden)
```

6. 1. انشاء و فتح الملفات باستخدام Open()

في الكود التالي ستجد اننا نستخدم الدالة Open() مع الخاصية FileMode.OpenOrCreate وهي ما تفيد انه لو وجدت الملف قم بفتحه ، لو لم تجده قم بانشاءه:

كود C#

```
FileInfo f2 = new FileInfo(@"C:\Test2.txt");
FileStream fs2 = f2.Open(FileMode.OpenOrCreate, FileAccess.ReadWrite,
FileShare.None);
```

كود VB.NET

```
Dim f2 As New FileInfo("C:\Test2.txt")
Dim fs2 As FileStream = f2.Open(FileMode.OpenOrCreate, FileAccess.ReadWrite,
FileShare.None)
```

هناك حالات أخرى أيضاً لفتح وانشاء الملفات:

الحالة	الوصف
CreateNew()	انشأ ملف جديد مباشرة ، لو وجدته قم بعمل خطأ IOException
Create()	قم بانشاء ملف ولو وجدته قم بانشاءه فوق الموجود
Open()	افتح الملف ولو لم تجده قم بعمل خطأ FileNotFoundException
OpenOrCreate()	فتح الملف لو كان موجود أو انشأه لو لم يكن موجوداً
Truncate()	افتح الملف وامسح كافة محتوياته
Append()	قم بفتح الملف وانتقل لآخره لبدء عملية الكتابة ، لو لم يكن الملف موجوداً قم بفتح واحد جديد والبدء بالكتابة من أوله ، عموماً هي لا تظهر في حالة الفتح باستخدام Open وإنما فقط في حالة OpenWrite

الجدول 13. 5. الدوال الخاصة بالفئة FileInfo.

ال FileAccess ايضاً له ثلاث حالات مجموعة في هذا ال enum :

كود	C#

```
public enum FileAccess
{
    Read,
    Write,
    ReadWrite
}
```

كود	VB.NET

```
Public Enum FileShare
    None
    Read
    Write
    ReadWrite
End Enum
```

6 .2. فتح و انشاء الملفات باستخدام OpenRead() و OpenWrite()

مثل ما سبق ، عدا انك ستكون ملزماً بنوع واحد فقط من العمليات.

6 .3. فتح الملفات باستخدام OpenText()

الفارق الوحيد هو ان القيمة المعادة تكون من نوع StreamReader وليس FileStream، بالشكل التالي مثلاً:

كود	C#

```
FileInfo f5 = new FileInfo(@"C:\boot.ini");
StreamReader sreader = f5.OpenText();
```

كود	VB.NET

```
Dim f5 As New FileInfo("C:\boot.ini")
Dim sreader As StreamReader = f5.OpenText()
```

6. 4. الفتح باستخدام ()CreateText و ()AppendText

مثل السابقة ، الفارق فقط يمكن ان القيمة المعادة ستكون من نوع StreamWriter.

7. التعامل مع الفئة File

نفس ما ذكرنا في حالة المجلدات ، لن تكون مجبراً على تعريف نسخة منها نظراً لإنها static بل يمكنك العمل عليها مباشرة.

هناك دوال أخرى جديدة في الفئة لعمليات القراءة والكتابة هي:

الدالة	الوصف
ReadAllBytes()	قراءة البيانات على شكل جدول array of bytes
ReadAllLines()	قراءة البيانات على شكل مصفوفة من الاسطر
ReadAllText()	قراءة جميع البيانات ككتلة واحدة
WriteAllBytes()	كتابة byte by byte
WriteAllLines()	الكتابة على شكل اسطر
WriteAllText()	الكتابة ككتلة واحدة

الجدول 13. 6. بعض الدوال الخاصة بالفئة File.

ويمكن استخدامهم بالشكل التالي مثلاً - من كتاب ProCSharp 2008 :

C# كود

```
InitializeComponent();
string[] myTasks = {
                    "Fix bathroom sink", "Call Dave",
                    "Call Mom and Dad", "Play Xbox 360"
                   };
// Write out all data to file on C drive.
File.WriteAllLines(@"C:\tasks.txt", myTasks);
// Read it all back and print out.
foreach (string task in File.ReadAllLines(@"C:\tasks.txt"))
{
    Console.WriteLine("TODO: {0}", task);
}
```

VB.NET كود

```
Dim myTasks As String() = {"Fix bathroom sink", "Call Dave", "Call Mom and
Dad", "Play Xbox 360"}
' Write out all data to file on C drive.
File.WriteAllLines("C:\tasks.txt", myTasks)
' Read it all back and print out.
For Each task As String In File.ReadAllLines("C:\tasks.txt")
    Console.WriteLine("TODO: {0}", task)
Next
```

معلومة اضافية

في net. ، يمكن للفئة إذا كانت تنجز الواجهة IDisposable ان نقوم بتعريفها في مكان ومن ثم حذفها خارج هذا النطاق بالشكل التالي مثلاً:

C# كود

```
using (CarClass newCare)
{
    // do operations
}
```

VB.NET كود

```
Using newCare As CarClass
' do operations
End Using
```

لذا سنحاول استخدامها ايضاً مع الملفات حتى لا تبقى معلقة في الذاكرة بالشكل التالي مثلاً:

C# كود

```
FileInfo f = new FileInfo(@"C:\test.txt");
using (StreamReader reader = f.OpenText())
{
    // code here
}
```

كود	VB.NET

```
Dim f As New FileInfo("C:\test.txt")
Using reader As StreamReader = f.OpenText()
' code here
End Using
```

8. Stream

ال Stream يقصد به تدفق او نقل البيانات ما بين مصدر ومستقبل سواء كان ذلك بين ملفين او بين جهازين على الشبكة أو طابعة أو خلافه ، حيث يتم نقل البيانات في الغالب على شكل sequance of bytes حتى نهاية الملف أو البيانات.

في net. ، هناك الفئة System.IO.Stream والتي تعتبر الفئة الأم التي سنتعامل مع غالب فئاتها لاحقاً ، تحتوي هذه الفئة على الدوال والخصائص الرئيسية التالية :

الخاصية أو الدالة	الوصف
CanRead, CanWrite	خصائص تتيح لك معرفة إذا كانت عملية ال Stream هذه تقبل عملية الكتابة أو القراءة وغيرها
Close()	اغلاق العملية وكل ما يتعلق بها من ملفات وخلافه
Flush()	تحديث بيانات المستقبل بالبيانات الموجودة حالياً في Buffer، لو لم يكن هذه العملية تدعم وجود Buffer فهذه الدالة لا تقوم بأي شيء في الواقع
Length	خاصية تعيد حجم ال stream بالبايت
Position	تحدد المكان في ال stream
Read(), ReadByte()	قراءة بايت او مجموعة منbytes
Seek()	وضع المؤشر في مكان جديد في هذا الstream
SetLength()	تحديد طول ال stram الحالي
Write(), WriteByte()	كتابة بايت او مجموعة من ال bytes

الجدول 13. 7. بعض الدوال الخاصة بالفئة Stream.

8. 1. الفئة FileStream

هذه الفئة التي تطبق الفئة القاعدية abstract class السابق تختص فقط بالتعامل مع ال streaming مع الملفات.

- المثال التالي يوضح الكتابة في عدة أماكن من الملف ومن ثم قراءة البيانات المكتوبة قبل الكتابة فقط نحتاج لتحويل الرسالة إلى bytes لذا لا تنس هذه الخطوة.

كود | C#

```
private void filestreamexample(string msg, string beforeendmsg)
{
    using (FileStream f1 = File.Open(@"C:\test.txt",
    FileMode.Create))
    {
        byte[] msgArray = Encoding.Default.GetBytes(msg);
        byte[] beforeendmsgArray = Encoding.Default.GetBytes(beforeendmsg);
        f1.Write(msgArray, 0, msgArray.Length);
        f1.Position = fStream.Lenght - 10;
        f1.Write(beforeendmsgArray, 0, msgAsByteArray.Length);

        // view total message.
        long totalLenght = msgArray.Length + msgAsByteArray.Length;
        byte[] filemsg = new byte[totalLenght];
        for (int i = 0; i < totalLenght; i++)
            filemsg[i] = (byte)f1.ReadByte();

        Console.WriteLine(Encoding.Default.GetString(filemsg));
    }
}
```

كود | VB.NET

```
Private Sub filestreamexample(ByVal msg As String, ByVal beforeendmsg As
String)
    Using f1 As FileStream = File.Open("C:\test.txt", FileMode.Create)
        Dim msgArray As Byte() = Encoding.[Default].GetBytes(msg)
        Dim beforeendmsgArray As Byte() =
Encoding.[Default].GetBytes(beforeendm
        f1.Write(msgArray, 0, msgArray.Length)
        f1.Position = fStream.Lenght - 10
        f1.Write(beforeendmsgArray, 0, msgAsByteArray.Length)
        ' view total message.
        Dim totalLenght As Long = msgArray.Length + msgAsByteArray.Length
        Dim filemsg As Byte() = New Byte(totalLenght - 1) {}
        For i As Integer = 0 To totalLenght - 1
            filemsg(i) = CByte(f1.ReadByte())
        Next
        Console.WriteLine(Encoding.[Default].GetString(filemsg))
    End Using
End Sub
```

8. 2. التعامل مع الفئات المشتقة

ذكرنا في أول دروسنا في موضوع الادخال والإخراج أن لدينا عدة انواع من القراءة والكتابة في stream ذكرنا منها StreamReader, StreamWriter و StringReader, StringWriter و BinaryWriter, BinaryReader ، وهم ما سنبدأ في التعرف عليهم الآن.

8. 3. StreamReader, StreamWriter

الفئات الأشهر والأكثر استخداماً من بين الفئات الثلاث السابقة ، تحتوي على الدوال الاساسية لعملية الكتابة والقراءة التالية:

StreamWriter

الدالة أو الخاصية	الوصف
Close()	اغلاق
Flush()	مسح كافية محتويات الbuffer
NewLine	سطر جديد
Write()	كتابة عادية
WriteLine()	كتابة مع سطر جديد

الجدول 13. 8. بعض الدوال الخاصة بالفئة StreamWriter.

StreamReader

تحتوي بالاضافة إلى الدوال الثلاث الأولى في الفقرة السابقة:

الدالة أو الخاصية	الوصف
Peek()	قراءة الحرف التالي مباشرة للمكان الحالي دون تغيير مكان المؤشر
Read()	القراءة
ReadBlock()	قراءة مجموعة من الداتا ووضعها في الbuffer
ReadLine()	قراءة سطر كامل
ReadToEnd()	قراءة من مكان المؤشر حتى النهاية

الجدول 13. 9. بعض الدوال الخاصة بالفئة StreamReader.

أمثلة

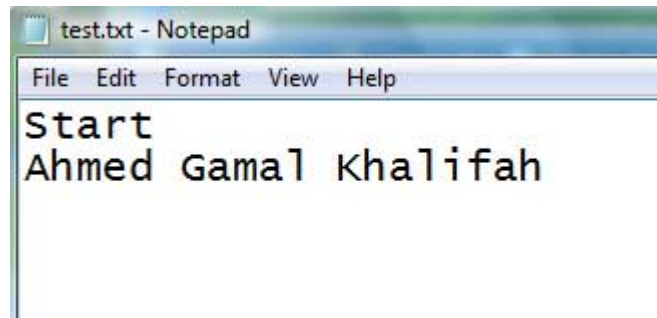
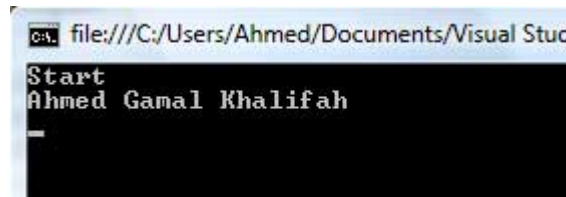
مثال يجمع عمليتي القراءة والكتابة إلى ملف نصي:

كود　C#

```
private void write(string msg)
{
    using (StreamWriter w1 = File.CreateText("test.txt"))
    {
        w1.Write("Start");
        w1.Write(writer.NewLine);
        w1.WriteLine(msg);
    }
}
private string read()
{
    using (StreamReader s1 = File.OpenText("test.txt"))
    {
        string input = null;
        string msg = "";
        while ((input = sr.ReadLine()) != null)
        {
            Console.WriteLine(input);
            msg += input;
        }
        return msg;
    }
}
```

كود　VB.NET

```
Private Sub write(ByVal msg As String)
    Using w1 As StreamWriter = File.CreateText("test.txt")
        w1.Write("Start")
        w1.Write(writer.NewLine)
        w1.WriteLine(msg)
    End Using
End Sub
Private Function read() As String

    Using s1 As StreamReader = File.OpenText("test.txt")
        Dim input As String = Nothing
        Dim msg As String = ""
        While (input = sr.ReadLine()) IsNot Nothing
            Console.WriteLine(input)
            msg += input
        End While
        Return msg
    End Using

End Function
```

ناتج عملية الكتابة في الملف:

وناتج عملية القراءة على ال:Console

ملاحظة

ناتج المتغير النصي سيكون بدون علامة السطر الجديد والذي سينتج عن الدالة

8. 4. StringWriter, StringReader

كما ذكرنا سابقاً لا يوجد اي فارق عما سبق ، فقط الناتج يعود في string وهو ما يمكنك من عملية بالشكل التالي مثلاً:

كود C#

```
StringBuilder stbuilder = strWriter.GetStringBuilder();
stbuilder.Remove(0, 10);
stbuilder.Insert(0, "Ahmed Gamal");
```

كود VB.NET

```
Dim stbuilder As StringBuilder = strWriter.GetStringBuilder()
stbuilder.Remove(0, 10)
stbuilder.Insert(0, "Ahmed Gamal")
```

8. 5. BinaryWriter, BinaryReader

تمكنك من الكتابة والقراءة والتخزين في ملفات binary، مشتقة من الفئة System.Object وتحتوي على أغلب الدوال الموجودة في الاربع فئات السابقة. اضافة إلى الدالة ()ReadXXXX حسب نوع المحتويات ، هذا المثال للقراءة والكتابة باستخدام هذه الفئة:

كود　C#

```
FileInfo f = new FileInfo("Binary.dat");
using (BinaryWriter bwriter = new BinaryWriter(f.OpenWrite()))
{
    double dbl = 1234.67;
    long lng = 34567000;
    string str = "A, B, C";
    // Write different data.
    bwriter.Write(dbl);
    bwriter.Write(lng);
    bwriter.Write(str);
}

// Reading using ReadXXX;
using (BinaryReader breader = new BinaryReader(f.OpenRead()))
{
    Console.WriteLine(breader.ReadDouble());
    Console.WriteLine(breader.ReadInt64()); // == long as i think , i am not
sure.
    Console.WriteLine(breader.ReadString());
}
```

كود	VB.NET

```
Dim f As New FileInfo("Binary.dat")
Using bwriter As New BinaryWriter(f.OpenWrite())
    Dim dbl As Double = 1234.67
    Dim lng As Long = 34567000
    Dim str As String = "A, B, C"

    ' Write differnt data.
    bwriter.Write(dbl)
    bwriter.Write(lng)
    bwriter.Write(str)
End Using

' Reading using ReadXXX;
Using breader As New BinaryReader(f.OpenRead())
    Console.WriteLine(breader.ReadDouble())
    Console.WriteLine(breader.ReadInt64())
    ' == long as i think , i am not sure.
    Console.WriteLine(breader.ReadString())
End Using
```

9. FileSystemWatcher

تعتبر هذه الفئة مفيدة جداً في حالة رغبتنا في مراقبة سلوك ملف معين ومعرفة اي تغيير يطرأ عليها من التغييرات الموجودة في الenum System.IO.NotifyFilters والذي يراقب اي من التغييرات التالية:

كود	C#

```
public enum NotifyFilters
{
    Attributes, CreationTime,
    DirectoryName, FileName,
    LastAccess, LastWrite,
    Security, Size,
}
```

كود	VB.NET

```
Public Enum NotifyFilters
    Attributes
    CreationTime
    DirectoryName
    FileName
    LastAccess
    LastWrite
    Security
    Size
End Enum
```

طبعاً بما اننا نتحدث عن events وتغييرات ، سنحتاج فوراً لتعريف دوال يتم تنفيذها مرتبطة بالاحداث ، اي دالة تستخدم لقراءة احداث مثل التعديل والتغيير وخلافه لا بد ان تكون على شكل delegate التالي:

كود	C#

```
void MyNotificationHandler(object source, FileSystemEventArgs e)
{

}
```

كود	VB.NET

```
Private Sub MyNotificationHandler(ByVal source As Object, ByVal e As
FileSystemEventArgs)

End Sub
```

أما حدث تغيير الاسم لا بد أن يتبع للشكل التالي:

كود	C#

```
void MyNotificationHandler(object source, RenamedEventArgs e)
{

}
```

كود	VB.NET

```
Private Sub MyNotificationHandler(ByVal source As Object, ByVal e As
RenamedEventArgs)

End Sub
```

الآن سنقوم بتعريف برنامج عادي ، يقوم بمراقبة حالة الملفات ، طبعاً ولإننا نرغب في ان نرى نتائج البرنامج فلا بد ان نجعل البرنامج يعمل حتى يضغط المستخدم حرف q مثلاً ، لو كنا في Windows Forms كان بامكاننا المراقبة مباشرة طبعاً ، في مثالنا هذا سنراقب كافة الملفات النصية في القرص الصلب \:C لحدثي الاضافة والحذف:

كود | C#

```
FileSystemWatcher watcher = new FileSystemWatcher();
// monitor files at:
watcher.Path = @"c:\";
// monitor files when
watcher.NotifyFilter = NotifyFilters.LastAccess | NotifyFilters.LastWrite |
NotifyFilters.FileName | NotifyFilters.DirectoryName;

// watch files of type
watcher.Filter = "*.txt";
// watch events:
watcher.Created += new FileSystemEventHandler(OnChanged);
watcher.Deleted += new FileSystemEventHandler(OnChanged);


watcher.EnableRaisingEvents = true;
Console.WriteLine("Press 'q' to quit app.");
while (Console.Read() != 'q') ;
```

كود | VB.NET

```
Dim watcher As New FileSystemWatcher()
' monitor files at:
watcher.Path = "c:\"
' monitor files when
watcher.NotifyFilter = NotifyFilters.LastAccess Or NotifyFilters.LastWrite Or
NotifyFilters.FileName Or NotifyFilters.DirectoryName

' watch files of type
watcher.Filter = "*.txt"
' watch events:
AddHandler watcher.Created, AddressOf OnChanged
AddHandler watcher.Deleted, AddressOf OnChanged


watcher.EnableRaisingEvents = True
Console.WriteLine("Press 'q' to quit app.")
While Console.Read() <> "q"c

End While
```

كما لاحظت، عندما يحدث أي حدث فإننا نقوم باستدعاء الدالة OnChanged ، يمكن تخصيص دالة لكل حدث أو كما تحب لكن هذا فقط مثال ، يمكننا كتابة الدالة OnChanged بالشكل التالي مثلاً - عرض الملف والتغيير ووقت التغيير:

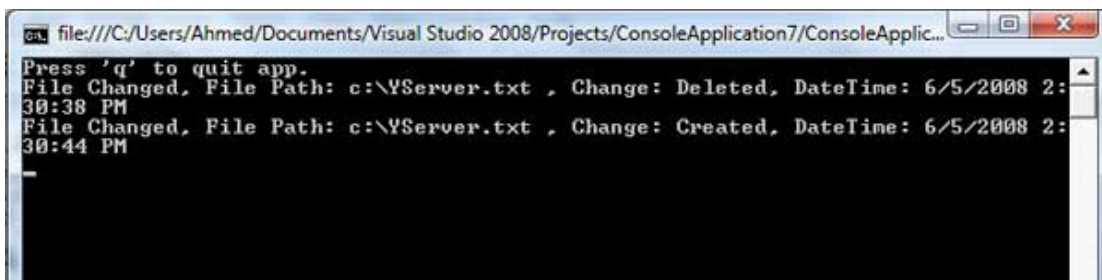
كود	C#

```
static void OnChanged(object source, FileSystemEventArgs e)
{
    Console.WriteLine("File Changed, File Path: {0} , Change: {1}, DateTime:
{2}", e.FullPath, e.ChangeType, DateTime.Now.ToString());

}
```

كود	VB.NET

```
Private Shared Sub OnChanged(ByVal source As Object, ByVal e As
FileSystemEventArgs)
    Console.WriteLine("File Changed, File Path: {0} , Change: {1}, DateTime:
{2}", e.FullPath, e.ChangeType, DateTime.Now.ToString())
End Sub
```

والآن جرب البرنامج...

الصورة التالية نتاج متوقع لحذف ملف من سواقة السي واعادته مرة أخرى) من سواقة السي مباشرة (:

الصورة 13. 2. نتائج تنفيذ الشفرة.

10. Object Serialization

سنتعرف في هذا الدرس سريعاً عن مفهوم ال Object Serialization.

10 .1. التعامل مع ال Serialization

عمل الفئة الخاصة بك لتكون Serializable:

كل ما في الأمر أن تضع الكلمة[Serializable] اعلى اسم الفئة بالشكل التالي مثلاً: التغيير:

كود | C#

```
[Serializable]
public class serial
{
}
```

كود | VB.NET

```
<Serializable()> _
Public Class serial

End Class
```

ماذا استفيد من كون الفئة الخاصة بي Serializable؟

معظم الفئات الاساسية تتيح لك كونها Serializable للاستفادة من بعض الخصائص مثل الكتابة المباشرة إلى ملفات القرص الصلب كما تعلمنا سابقاً ، هذا المثال:

كود | C#

```
serial sample = new serial();
using (Stream fs1 = new FileStream("data.txt",
FileMode.Create, FileAccess.Write, FileShare.None))
{
    binFormat.Serialize(fw1, sample);
}
```

كود | VB.NET

```
Dim sample As New serial()
Using fs1 As Stream = New FileStream("data.txt", FileMode.Create,
FileAccess.Write, FileShare.None)
    binFormat.Serialize(fw1, sample)
End Using
```

ماذا أيضاً ؟

هناك عدد كبير من الكائنات التي قد لا تقبل التعامل مع الفئات الخاصة بك إلا لو كانت Serializable، منها ال View State في صفحات الASP.net.

هناك العديد من الطرق لعمل Serialization لكائناتك ، منها:

BinaryFormatter

عليك أولا اضافة مرجع إلى المجمع

System.Runtime.Serialization.Formatters.Binary

كود　C#

```
BinaryFormatter binFormatter = new BinaryFormatter();
using (Stream fs1 = new FileStream("data.txt",FileMode.Create,FileAccess.Write,
FileShare.None))
{
    binFormat.Serialize(fs1, myobject);
}
```

كود　VB.NET

```
Dim binFormatter As New BinaryFormatter()
Using fs1 As Stream = New FileStream("data.txt", FileMode.Create,
FileAccess.Write, FileShare.None)
    binFormat.Serialize(fs1, myobject)
End Using
```

ولعمل Deserializing:

كود　C#

```
BinaryFormatter binFormatter = new BinaryFormatter();
using (Stream fs1 = File.OpenRead("data.txt"))
{
    newobject car =
    (objectclass)binFormatter.Deserialize(fs1);
}
```

كود　VB.NET

```
Dim binFormatter As New BinaryFormatter()
Using fs1 As Stream = File.OpenRead("data.txt")
    Dim car As newobject = DirectCast(binFormatter.Deserialize(fs1),
objectclass)
End Using
```

10 .2 XmlSerializer

يعد ال XML النوع الأحدث في عالم تخزين البيانات نظراً لإنه يجمع بين امكانيات قواعد البيانات من حيث البحث وخلافه ويجمع بين سهولة التعامل معها كما في الملفات النصية، الصيغة العامة لأي ملف XML بالشكل التالي:

كود | XML

```xml
<root>
  <Member>
    <Name>AHmed</Name>
    <Age>22</Age>
  </Member>
</root>
```

أحمد و 22 تسمى Element أما Name و Age تسمى Attribute .

لمعرفة المزيد حول XML يمكنك البدء من هنا:

رابط

http://www.w3schools.com/xml/xml_whatis.asp

العناصر الأساسية الموجودة في System.Xml.Serialization

العنصر	الوصف
XmlAttributeAttribute	يتم تحويل هذا العنصر على انهAttribute
XmlElementAttribute	يتم تحويل هذا العنصر على انهElement
XmlRootAttribute	لتحديد الRoot
XmlTextAttribute	يتم تحويلها لXML Text

الجدول 13 .10. بعض دوال مجال الأسماء System.Xml.Serialization.

فمثلاً لتحويل هذه الفئة إلى XML عن طريق عملية Serialization:

كود | C#

```csharp
class Members
{
    [XmlAttribute]
    string Name;
    [XmlAttribute]
    int Age;
}
```

VB.NET	كود

```
Class Members
    <XmlAttribute()> _
    Private Name As String

    <XmlAttribute()> _
    Private Age As Integer

End Class
```

والآن بعد اضافة اي نوع من البيانات على هذه الفئة ومن ثم عمل Serialization ستجد الناتج على شكل ملف XML السابق.

الباب

14

برمجة النوافذ في

.net 2008

Windows Forms

1. مقدمة

تعتبر الكلمة Viusal التي اصبحت متلازمة مع اغلب لغات البرمجة هي الحل السحري لمشاكل تصميم واجهات مناسبة للمستخدم ، في عصر ما قبل لغات البرمجة Visual كنت ستكون مجبر على رسم واجهاتك بالكود ، مع ما يعنيه ذلك من صعوبة وتأخير في وقت التنفيذ اضافة إلى انخفاض مستوى التصميم.

في net. جاءت المكتبة System.Windows.Forms لتقدم لك كل ما تحتاج إليه لتصميم واجهة مناسبة ، هناك أيضاً مجموعة دوال API تعرف باسم +GDI تمكنك من زيادة كفاءتك الرسومية من خلال مجموعة من الأوامر المخصصة للرسومات 2D، أخيراً ومع net. 3.0 جاءت لنا مايكروسوفت بتقنية Windows Presentation Foundation او ما تعرف اختصاراً باسم WPF لتزيد من كفاءة الرسوميات ضمن تطبيقات البرمجة لحد مرتفع جداً.

2. بناء Windows Forms بالكود

قبل أن نبدأ باستخدام المعالجات الجاهزة التي وفرتها لنا مايكروسوفت ضمن Visual Studio، يفضل ان نقوم بمحاولة بناء نموذج بعيداً عنها حتى نستطيع فهم كيفية عمل Windows Forms.

لذا قم بفتح المفكرة Notepad ، قم بانشاء ملف مثلاً باسم WindowsForms.cs، ومن ثم سنبدأ في كتابة الكود.

أولاً سنقوم بتعريف فئة مشتقة من الفئة Forms بالشكل التالي مثلاً :

كود	C#

```
class SimpleWindow : Form
{
}
```

كود	VB.NET

```
Class SimpleWindow
    Inherits Form
End Class
```

والآن سنقوم في حدث Main باستدعاء نسخة منه، لا تنسى اضافة مكتبات System.Windows.Forms، لذا سيكون الكود بالكامل بالشكل التالي:

كود	C#

```
using System;
using System.Windows.Forms;
namespace WindowsForms
{
    class Program
    {
        static void Main()
        {
            Application.Run(new SimpleWindow());
        }
    }
}
class SimpleWindow : Form
{
}
```

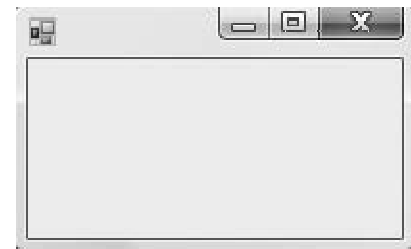
كود	VB.NET

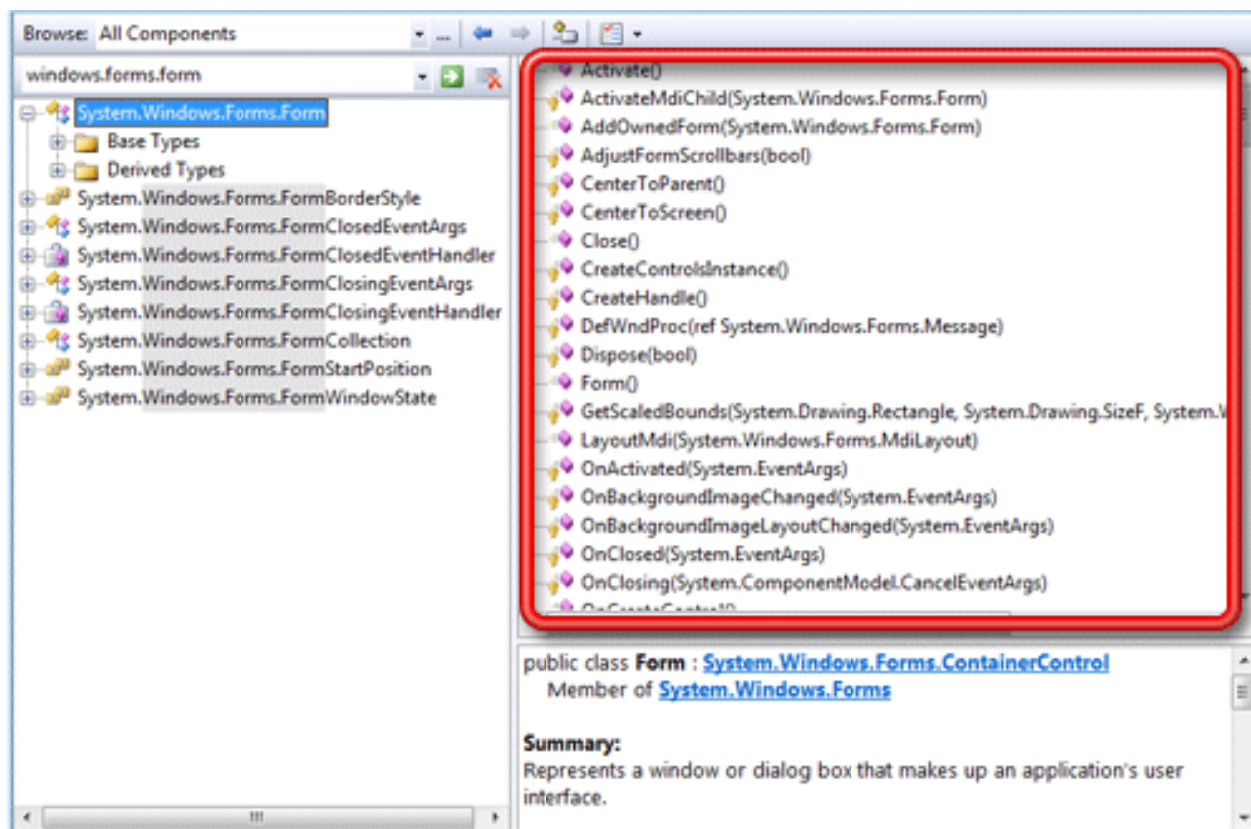
```
Imports System
Imports System.Windows.Forms
Namespace WindowsForms

    Class Program
        Private Shared Sub Main()
            Application.Run(New SimpleWindow())
        End Sub
    End Class

End Namespace
Class SimpleWindow
    Inherits Form

End Class
```

الآن فقط كل ما عيك هو فتح الشل Cmd الخاص بك ، وكتابة أمر مثل التالي:

كود	Shell

```
csc /target:winexe *.cs
```

طبعاً بعد الذهاب لمسار البرنامج، وبعد تعريف CSC لتكون جاهزة للاستخدام في أي مكان، سيكون الناتج شيئاً مثل هذا:

الصورة 14. 1. بناء نافذة بسيطة.

والآن بامكاننا التحكم في بعض الخصائص ، مثل الطول والعرض . الشفافية .... الخ من الخصائص أو حتى تنفيذ الأوامر التي يمكن الوصول إليها من خلال Object Browser بالشكل التالي مثلاً:

الصورة 14. 1. نافذة ال Object Explorer.

لذا نجد ان بامكاننا مثلاً تغيير الطول والعرض وتوسيطه ، لذا سيكون كود الفورم بالشكل التالي:

كود	C#

```
class SimpleForm : Form
{
    public SimpleForm(int height, int width, bool center)
    {
        Width = width;
        Height = height;
        if (center) CenterToScreen();
    }
}
```

كود	VB.NET

```
Class SimpleForm
    Inherits Form
    Public Sub New(ByVal height As Integer, ByVal width As Integer, ByVal
center As Boolean)
        Width = width
        Height = height
        If center = True Then
            CenterToScreen()
        End If
    End Sub
End Class
```

لكن لا تنس أنك في Run سوف تقوم بتمرير بارميترات إلى الفئة ، بالشكل التالي مثلاً:

كود	C#

```
static void Main()
{
    Application.Run(new MainWindow(200, 300, True));
}
```

كود	VB.NET

```
Private Shared Sub Main()
    Application.Run(New MainWindow(200, 300, [True]))
End Sub
```

لا تنس طبعاً انه بامكانك الحصول على هذه القيم من خلال Args كما تعلمنا سابقاً ...

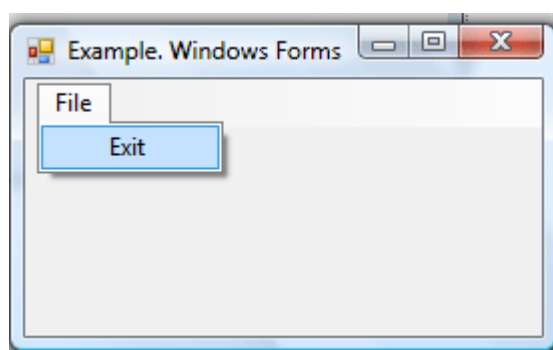
2 .1. اضافة أدوات بالكود

تعلمنا رسم الفورم بالكود، جاء الدور الآن على وضع الأدوات المختلفة على الفورم ، يتم ذلك بالواقع من خلال الأوامر التالية:

الدالة أو الخاصية	الوصف
Add()	لعمليات الاضافة
AddRange()	
Remove	لعمليات الحذف
RemoveAt()	
Clear()	لمسح الجميع
Count	عدد الادوات ، في الواقع هذه الخاصية مفيدة جداً عند محاولات المرور على الأدوات

أما الأدوات التي يمكنك اضافتها ، فجميعها موجود تحت الفئة System.Windows.Forms، حيث تجد كل الادوات التي تعودت على رؤيتها...

2 .2 اضافة القوائم

سنحاول الوصول للشكل التالي حيث يمكن الخروج بالضغط على زر Exit في القائمة:

الصورة 14 .2 اضافة أداة ال Menu للنافذة بالكود.

الكود سيكون بسيطاً ، سنقوم بتعريف MenuStrip للقائمة ومن ثم ToolStripMenuItem لعناصر القائمة:

كود	C#

```
private System.Windows.Forms.MenuStrip menuStrip1;
private System.Windows.Forms.ToolStripMenuItem fileToolStripMenuItem;
private System.Windows.Forms.ToolStripMenuItem exitToolStripMenuItem;
```

كود VB.NET

```
Private menuStrip1 As System.Windows.Forms.MenuStrip
Private fileToolStripMenuItem As System.Windows.Forms.ToolStripMenuItem
Private exitToolStripMenuItem As System.Windows.Forms.ToolStripMenuItem
```

الخطوة الثانية هي عمل دالة في ال Constructor وليكن اسمها BuildForm، بدلاً من وضع الاكواد مباشرة في الConstructor.

كود C#

```
public SimpleForm()
{
    Text="Example. Simple Form";
    BuildForm();
}
```

كود VB.NET

```
Public Sub New()
    Text = "Example. Simple Form"
    BuildForm()
End Sub
```

الآن سنقوم ببرمجة هذه الدالة، سنقوم بوضع اسماء للعناصر ونضيفها باستخدام Add للقائمة الرئيسية:

كود C#

```
fileToolStripMenuItem.Text = "&File";
menuStrip1.Items.Add(mnuFile);
exitToolStripMenuItem.Text = "E&xit";
menuStrip1.DropDownItems.Add(mnuFileExit);
```

كود VB.NET

```
fileToolStripMenuItem.Text = "File"
menuStrip1.Items.Add(mnuFile)
exitToolStripMenuItem.Text = "Exit"
menuStrip1.DropDownItems.Add(mnuFileExit)
```

الخطوة الثانية هي اضافة Handler لحدث الضغط لعنصر Exit بالشكل التالي:

كود C#

```
exitToolStripMenuItem.Click += new
System.EventHandler(this.exitToolStripMenuItem_Click);
```

كود	VB.NET

```
AddHandler exitToolStripMenuItem.Click, AddressOf exitToolStripMenuItem_Click
```

أخيراً سنعتمد القائمة الرئيسية لتكون القائمة الخاصة بال Form بالشكل التالي بعد اضافتها للفورم:

كود	C#

```
Controls.Add(this.mnuMainMenu);
MainMenuStrip = this.menuStrip1;
```

كود	VB.NET

```
Controls.Add(Me.mnuMainMenu)
MainMenuStrip = Me.menuStrip1
```

الآن سوف نقوم بكتابة الحدث الخاص بالضغط على زرExit:

كود	C#

```
private void exitToolStripMenuItem_Click(object sender, EventArgs e)
{
    Application.Exit();
}
```

كود	VB.NET

```
Private Sub exitToolStripMenuItem_Click(ByVal sender As Object, ByVal e As
EventArgs)
    Application.[Exit]()
End Sub
```

بهذا يكون الكود الاجمالي بالشكل التالي

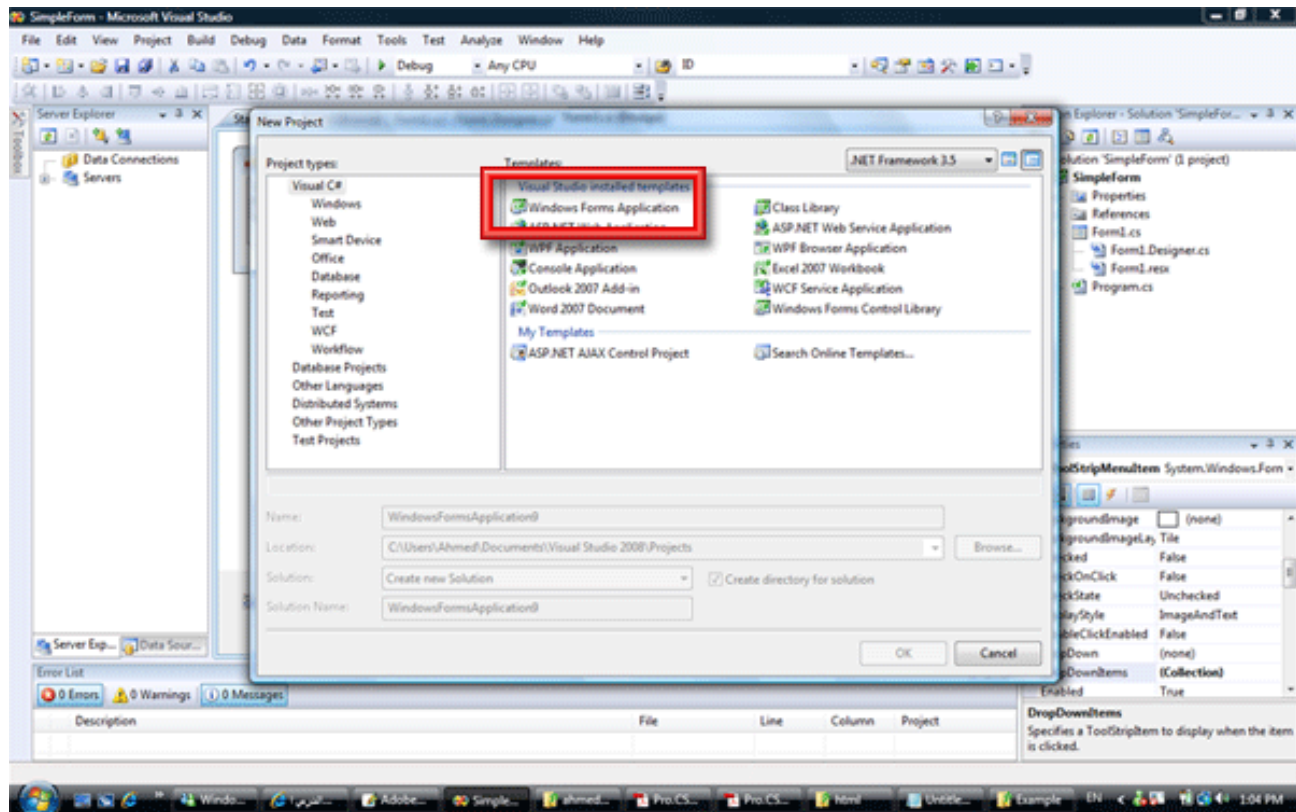
C# كود

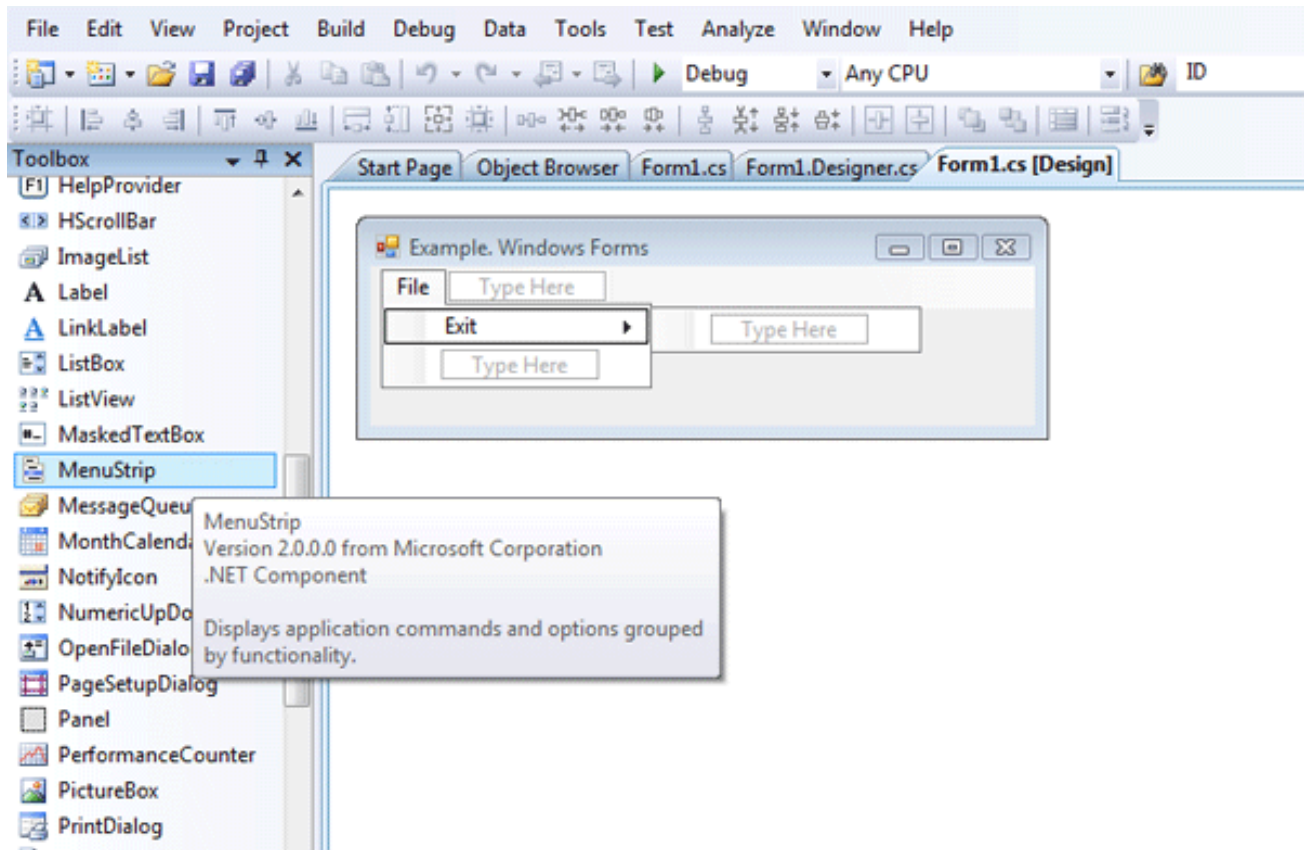
```
class SimpleForm : Form
{
    private System.Windows.Forms.MenuStrip menuStrip1;
    private System.Windows.Forms.ToolStripMenuItem fileToolStripMenuItem;
    private System.Windows.Forms.ToolStripMenuItem exitToolStripMenuItem;
    public SimpleForm()
    {
        Text = "Example. Simple Form";
        BuildForm();
    }
    private void BuildForm(){
        fileToolStripMenuItem.Text = "File";
        menuStrip1.Items.Add(mnuFile);
        exitToolStripMenuItem.Text = "Exit";
        menuStrip1.DropDownItems.Add(mnuFileExit);
        exitToolStripMenuItem.Click += new
System.EventHandler(this.exitToolStripMenuItem_Click);
        Controls.Add(this.mnuMainMenu);
        MainMenuStrip = this.menuStrip1;
    }
    private void exitToolStripMenuItem_Click(object sender, EventArgs e){
        Application.Exit();
    }
}
```

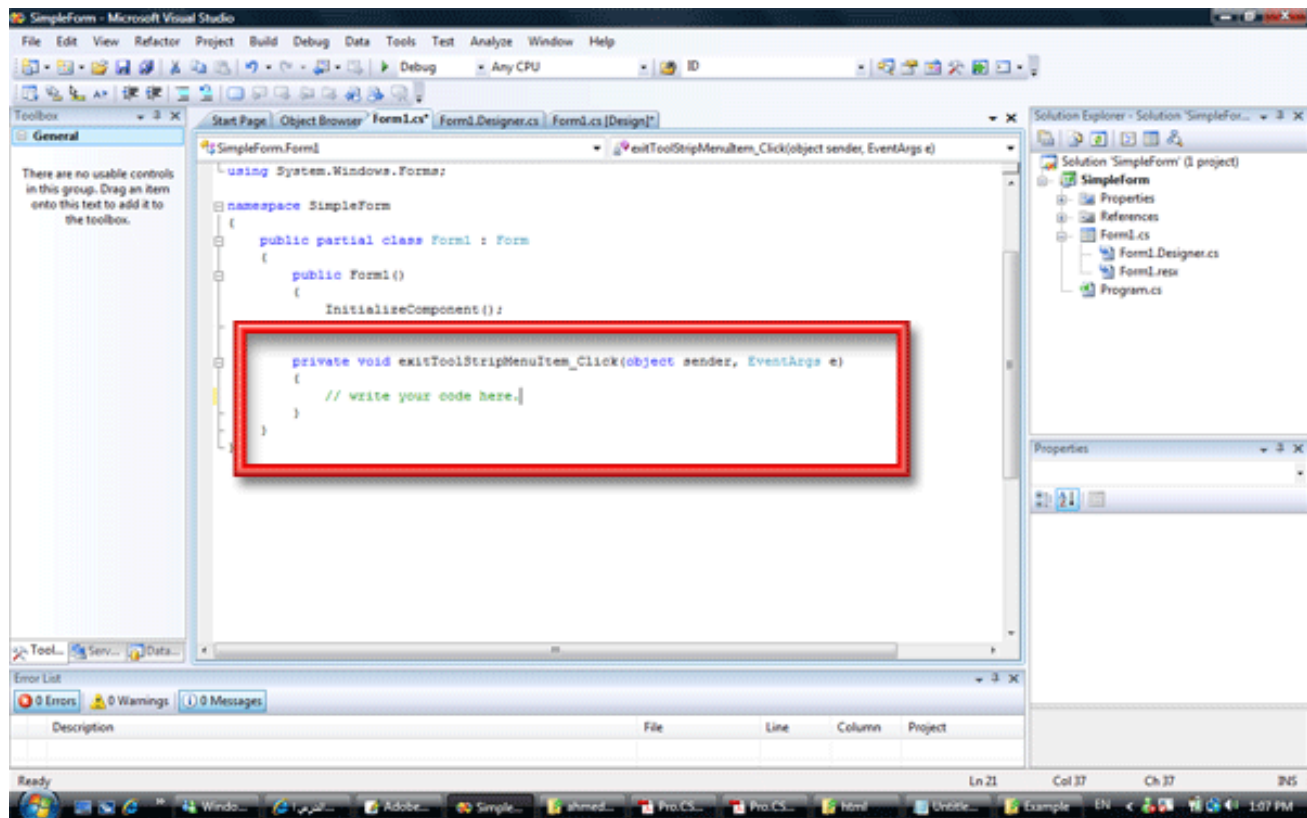
VB.NET كود

```
Class SimpleForm
    Inherits Form
    Private menuStrip1 As System.Windows.Forms.MenuStrip
    Private fileToolStripMenuItem As System.Windows.Forms.ToolStripMenuItem
    Private exitToolStripMenuItem As System.Windows.Forms.ToolStripMenuItem
    Public Sub New()
        Text = "Example. Simple Form"
        BuildForm()
    End Sub
    Private Sub BuildForm()
        fileToolStripMenuItem.Text = "&File"
        menuStrip1.Items.Add(mnuFile)
        exitToolStripMenuItem.Text = "E&xit"
        menuStrip1.DropDownItems.Add(mnuFileExit)
        AddHandler exitToolStripMenuItem.Click, AddressOf
exitToolStripMenuItem_Click
        Controls.Add(Me.mnuMainMenu)
        MainMenuStrip = Me.menuStrip1
    End Sub
    Private Sub exitToolStripMenuItem_Click(ByVal sender As Object, ByVal e As
EventArgs)
        Application.[Exit]()
    End Sub
End Class
```

3. انشاء فورم عن طريق Visual Studio .net

فقط افتح Visual Studio 2008، من قائمة File-New قم باختيار Windows Forms Application :

الصورة 14 .3. اضافة النوافذ بالاستعانة ببيئة التطوير Visual Studio 2008.

الآن من القائمة الجانبية Tool Box اختر MenuStrip ، ستجد الشكل التالي:

الصورة 14. 4. اضافة القوائم بالاستعانة ببيئة التطوير Visual Studio 2008.

الآن وكما فعلت قم بالتفرع كما تريد وتصميم القائمة، عند الضغط مرتين على العنصر Exit ستجد الأمر جاهزاً للكتابة بالشكل التالي:

الصورة 14. 5. توليد الكود اوتوماتيكيا بواسطة بيئة التطوير.

الآن اكتب الكود، ومن ثم اضغط على F5 وجرب.

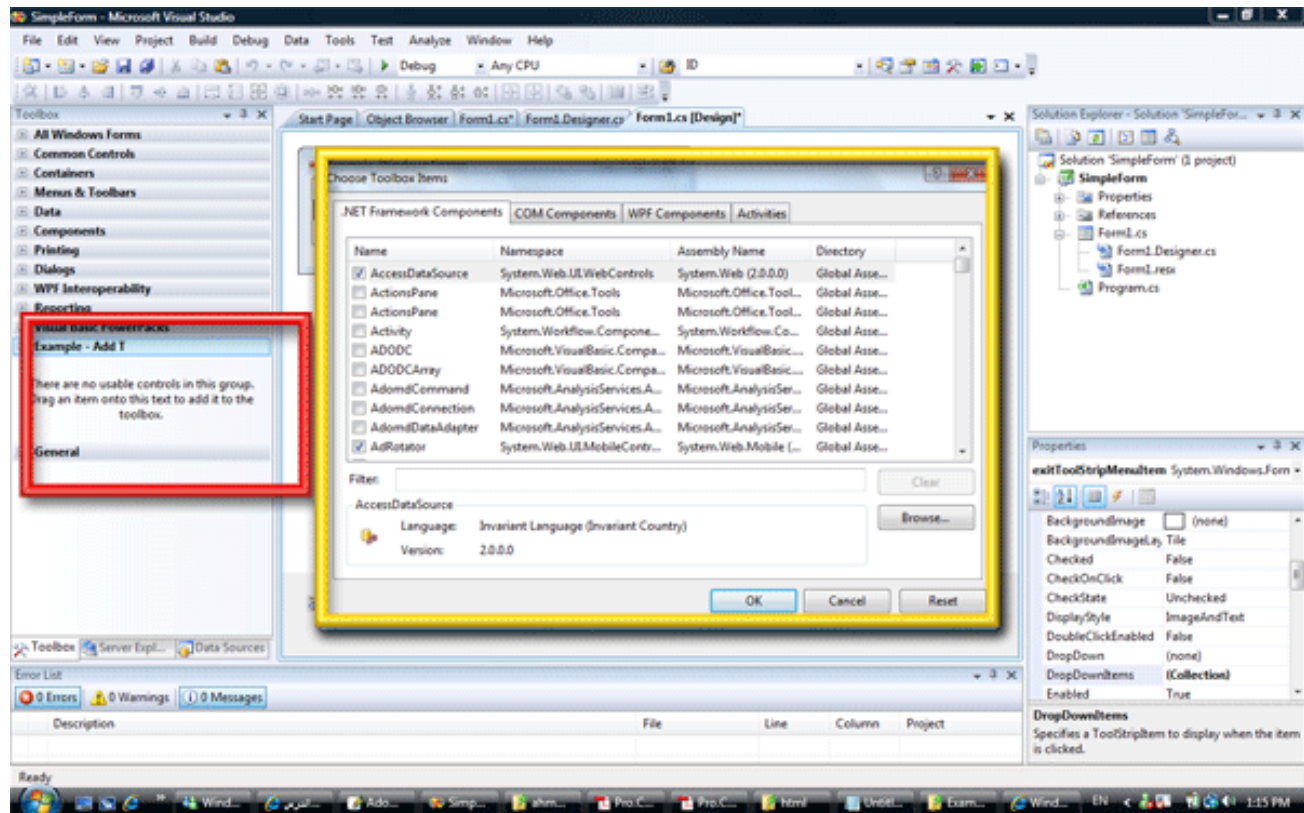
والآن ألا تتفق معي أن الوضع أصبح أسهل كثيراً....

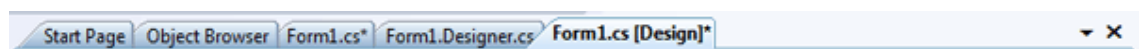
الآن ، سنتعرف على محتويات الواجهة الرئيسية.

في مرحلة سابقة من الدروس تعرفنا على بعض النقاط في واجهة Visual Studio، الآن سنواصل التعرف على النقاط الخاصة بالواجهة الرسومية.

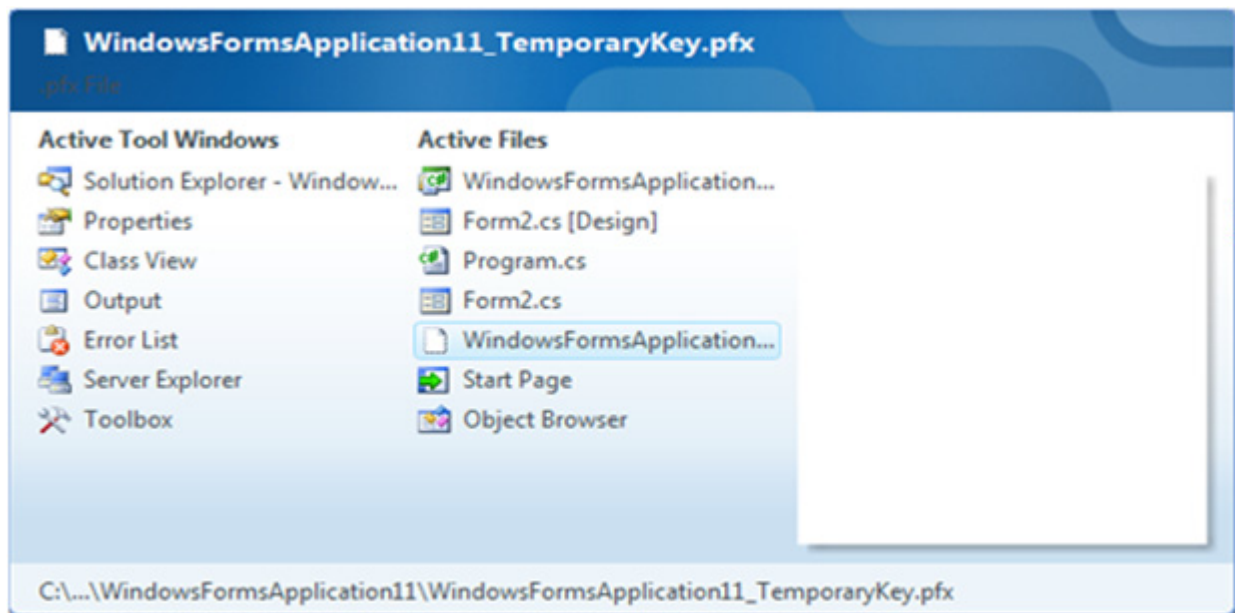
ToolBox

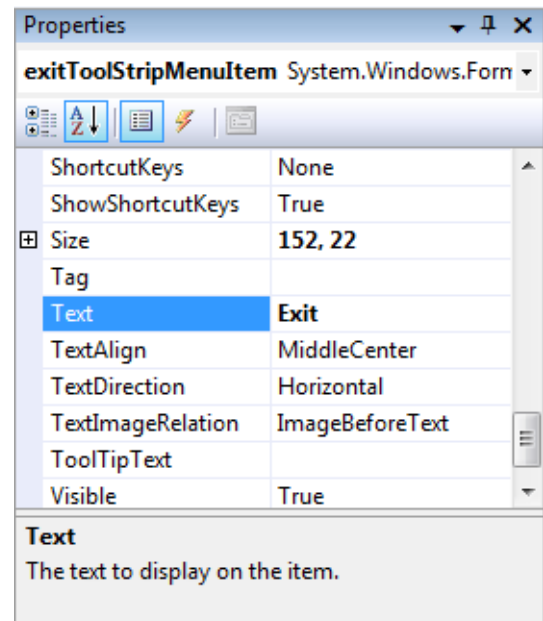
يحتوي على الأدوات الرئيسية التي نحتاج اليها ، لو لم يكن ظاهراً يمكننا اظهاره من View - ToolBox.

أهم الادوات التي نحتاج إليها موجودة تحت التبويب All Windows Forms ، مع ذلك ما زال بامكاننا اضافة تبويب جديد واضافة أدوات أخرى إليه ، عن طريق Add Tab ومن ثم Choose Item واختيار العنصر الذي ترغب فيه:

الصورة 14. 6. اضافة عناصر لقائمة الأدوات ToolBox.

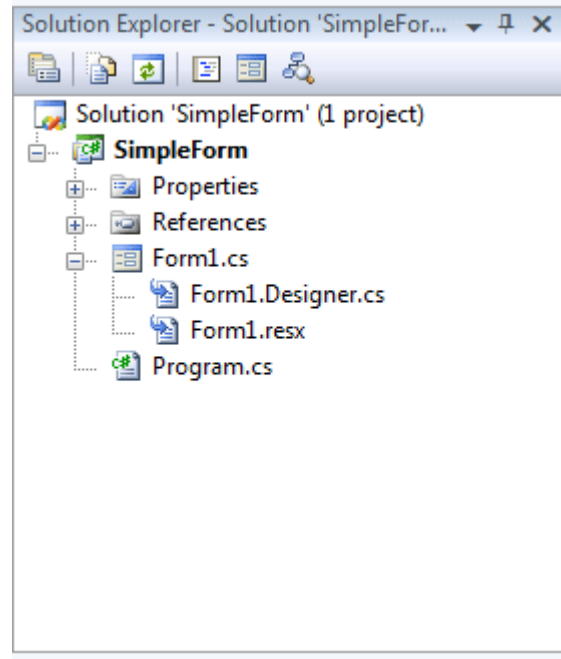
Tabs Group

الصورة 14. 7. شريط النوافذ.

يحتوي على جميع الملفات والمصادر التي قمت بفتحها، حيث يمكنك التنقل بينها.

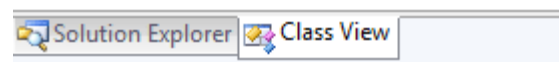
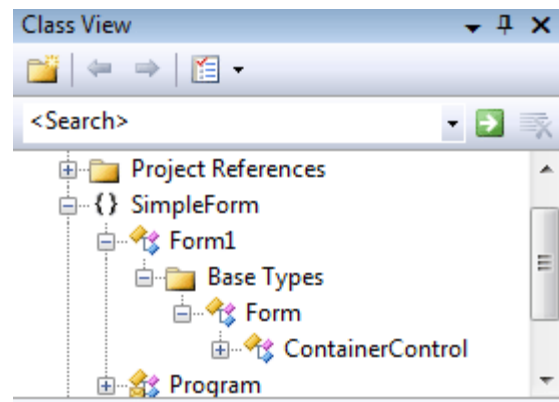
يمكنك الضغط على Ctrl+Tab للتبديل بين الشاشات المفتوحة بالشكل التالي مثلاً :

الصورة 14. 8. نافذة اختيار النوافذ.

Properties Windows

الصورة 14. 9. نافذة الخصائص.

تجد فيها الخصائص التي ترغب في تعديلها لأي عنصر يتم تحديده على الفورم ، هذه الصورة توضح خصائص احد عناصر القائمة.

Solution Explorer

الصورة 14. 10. نافذة متصفح المشاريع.

يمكنك من تصفح محتويات مشروعك من نوافذ Forms أو فئات Classes او حتى صور و Resources .

Class View

الصورة 14. 11. نافذة متصفح الكائنات.

لاستعراض الفئات الموجودة لديك وخصائصها ودوالها وخلافه.

كيف يعمل ال Visual Studio.net في انشاء النماذج ؟

بنفس الطريقة التي تعلمناها بالكود، ستجد لديك من قائمة Solution Explorer ثلاثة ملفات:

1- Form1.cs

سيمكنك من التحكم في الفورم يدوياً عن طريق الوضع [Design] ايضاً تجد الأكواد التي قمت بكتابتها في هذا الملف ، يقوم كثير من المبرمجين بالتبديل بين الوضعين بالتنقل من القائمة العلوية ، او الضغط مرتين على اي اداة لفتح الكود الخاص بها . يمكنك عمل ذلك بطريقة مثالية عن طريق الضغط على الفورم بزر الماوس الأيمن واختيار الوضع Design - Code أو حتى Class Diagram.

Form1.Designer.cs -2

هنا تجد الاكواد الخاصة بانشاء الأدوات والفورم ، لو قمت بفتحه ستجد اكواداً شبيهة بتلك التي استخدمناها في الجزء الأول من هذا الدرس:

```
            this.exitToolStripMenuItem.Size = new System.Drawing.Size(152, 22);
            this.exitToolStripMenuItem.Text = "Exit";
            this.exitToolStripMenuItem.Click += new System.EventHandler(this.exitTool
            //
            // Form1
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(375, 90);
            this.Controls.Add(this.menuStrip1);
            this.MainMenuStrip = this.menuStrip1;
            this.Name = "Form1";
            this.Text = "Example. Windows Forms";
            this.menuStrip1.ResumeLayout(false);
            this.menuStrip1.PerformLayout();
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        #endregion

        private System.Windows.Forms.MenuStrip menuStrip1;
        private System.Windows.Forms.ToolStripMenuItem fileToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem exitToolStripMenuItem;
    }
}
```

الصورة 14 .12. متصفح الأكواد.

Form1.rex -3

يحتوي على المصادر المستخدمة وخلافه.

4. مجال الأسماء System.Windows.Forms

4. 1. الفئات الرئيسية لعناصر System.Windows.Forms

الفورم هو العنصر الاساسي في هذه الفئة ، وهو مشتق من الفئات التالية:

System.Object

العنصر الاساسي لكل كائنات net.

System.MarshalByRefObject

لنتمكن من الوصول byref إلى الفورم

System.ComponentModel.Component

لثالث مشتق من ال Interface المسمى IComponent

System.Windows.Forms.Control

سنتناوله بالتفصيل لاحقاً.

System.Windows.Forms.ScrollableControl

من اجل استخدام ال Scrolls ، اي عنصر مشتق منها يمكن ان يتمتع بهذه الخاصية.

System.Windows.Forms.ContainerControl

من اجل جعله يمكن ان يحتوي على Controls أخرى ، اي عنصر مشتق منها يمكن ان يحتوي بداخله على عناصر أخرى.

System.Windows.Forms.Form

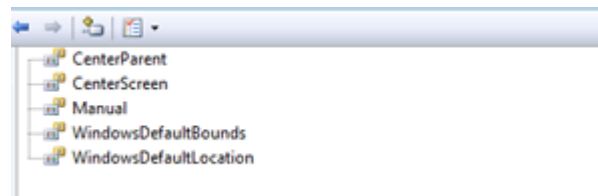
سنتناوله بالتفصيل لاحقاً.

ذكرنا أن ال Form مشتق من الفئة System.Windows.Forms.Form، لذا فإن جميع خصائص ودوال وأحداث هذه الفئة تنتقل له بالتبعية، وهي ما سنتعرف عليها الآن ...

4 .2. خصائص الفئة Form

الخاصية	الوصف
AcceptButton	تحديد الزر الذي يتم تنفيذ أوامره عند ضغط المستخدم على Enter
ActiveMDIChild	عندما يكون للفورم أبناء ، يتم هذا لتحديد الإبن النشط
IsMDIChildIsMDIContainer	لتحديد امكانية جعل هذا الفورم حاضن لفورمات أخرى
CancelButton	تحديد الزر الذي يتم تنفيذ أوامره عندما يضغط المستخدم على زرEsc
ControlBox	تحديد هل للفورم Control Box أم لا
FormBorderStyle	تحديد شكل حدود الفورم
Menu	تحديد القائمة الرئيسية للفورم
MaximizeBox	عرض زر التكبير والتصغير أم لا ، ستجد واحدة أخرى باسم Minimize لعرض زر وضعه في الشريط السفلي
ShowInTaskbar	تحديد العرض في الشريط السفلي للويندوز أم لا
StartPosition	تحديد نقطة البداية للفورم عند ظهوره لأول مرة ..
WindowState	تحديد حالة الفورم والتي يتم اختيارها من ال enumeration المسمى FormWindowState

الجدول 14. 1. بعض خصائص الفئة Form.

ويحتوي الـ enumeration المسمى FormStartPosition على العناصر التالية:

أما FormWindowState فيحتوي على :

4 .3. دوال الفئة Form

الدالة	الوصف
Activate()	لتنشيط الفورم
Close()	اغلاق الفورم
CenterToScreen()	وضع الفورم في منتصف الشاشة
LayoutMDI()	تحديد نظام عرض الشاشات الأبناء في حالة وجودها
ShowDialog()	عرض الفورم بصورة Dialog بحيث لا يمكن الرجوع لما خلفها إلا بعد اغلاقها

الجدول 14 .2. بعض دوال الفئة Form.

4 .4. أحداث الفئة Form

الحدث	الوصف
Activated	ينطلق هذا الحدث عندما يتم تنشيط الفورم سواء بأمر Activate
Closing	عندما يبدأ حدث الإغلاق
Closed	عندما ينتهي حدث الإغلاق
Deactivate	عندما يذهب التحكم إلى شيء آخر غير الفورم
Load	عندما يتم تحميل الفورم ، ولكن ينطلق هذا الحدث قبل ظهور الفورم على الشاشة أصلاً - لتجنب واحد من أكثر الأخطاء شيوعاً-
MDIChildActive	عند تحديد واحد من الشاشات الأبناء في حالة وجودها

الجدول 14 .3. بعض أحداث الفئة Form.

كما ذكرنا ايضاً سابقاً ، فإن الفورم مشتق من الفئة Controls ، ولذا فهو يتمتع بكافة مميزاتها والتي سنستعرضها هنا.

5. الفئة Controls

5. 1. خصائص الفئة Controls

الخاصية	الوصف
BackColor	لون الخلفية
ForeColor	لون النص الداخلي
BackgroundImage	خلفية صورة
Font	الخط وحجمه ونوعه وخلافه
Cursor	نوع المؤشر الذي يظهر عند العبور فوق الأداة أو الفورم
Anchor	للتعامل مع خاصية ال Dock والتي تثبت مكان الادوات على الفورم مهما تغير حجم الفورم
AutoSize	حجم تلقائي للاداة حسب محتوياتها
Top	موضع الاداة او الفورم من الحد العلوي للاب
Left	موضع الاداة او الفورم من الحد الأيسر للأب
Bottom	موضع الاداة او الفورم من الحد الأسفل للأب
Right	موضع الاداة او الفورم من الحد الأيمن للأب
Bounds	الحدود
ClientRectangle	من مربع
Height	الطول
Width	العرض
Enabled	تحديد امكانية التحكم بالأداة او الفورم من عدمه

Visible	الظهور والاختفاء
ModifierKeys	قراءة حالة ال ModifierKeys مثل ال Alt و Ctrl وShift
MouseButtons	معرفة الزر المضغوط من الماوس (أيمن - ايسر – المنتصف)
TabIndex	تحدد ترتيب العناصر للانتقال بينها بواسطة زرTab
TabStop	منع الوصول للعنصر من خلال زرTab
Opacity	درجة الشفافية ما بين 0 و 1
Text	النص المعروض داخل الاداة
Controls	تمكنك من الوصول للاداوات الداخلية في حالة كان العنصر قادراً على استيعاب عناصر داخله

الجدول 14. 4. خصائص الفئة Control.

5 .2. أحداث الفئة Controls

الحدث	الوصف
Click	الضغط بالماوس
DoubleClick	ضغطتين بالماوس
MouseEnter	دخول الماوس إلى نطاق الأداة
MouseLeave	خروج الماوس من نطاق الاداة
MouseDown	ضغط زر الماوس لاسفل
MouseUp	رفع الاصبع عن ضغطة زر الماوس
MouseMove, MouseHover	عبور الماوس فوق نقطة .
MouseWheel	لضغط على الWheel
KeyPress	ضغط زر من الكيبورد

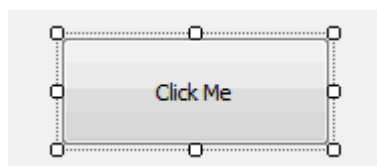
KeyUp	ضغط زر لاسفل
KeyDown	رفع الاصبع عن الزر المضغوط
DragDrop	سحب اداة فوق الاداة الحالية
DragEnter	دخول اداة لمجال اداة أخرى
DragLeave	خروج
DragOver	عبور فوق الأداة
Paint	حدث الرسم

الجدول 14. 5. أحداث الفئة Control.

6. أدوات System.Windows.Forms.Control

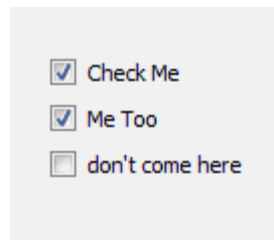
ذكرنا في الدرس السابق بإننا سنقوم باستعراض الأدوات ، أول نقطة عليك معرفتها هي أن كل الأدوات مشتقة من الفئة Controls لذا فإن أي أداة ستتمتع بالخصائص والأحداث التي قمنا بشرحها في الدرس السابق مباشرة.

الآن سنستعرض بعض الأدوات الرئيسية لدينا.

6. 1. Button

زر أمر عادي جداً ، يحتوي على بعض الخصائص مثل:

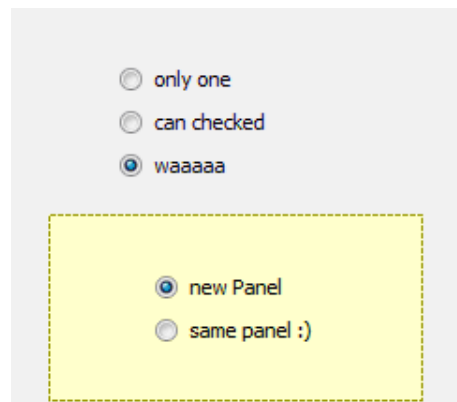
FlatStyle: تحديد مظهر زر الأمر.

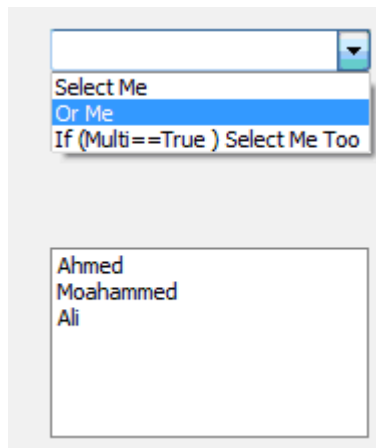
TextAlign: موضع النص من زر الأمر.

6. 2. CheckBox

زر الاختيار المتعدد ، يمكننا اختيار عدة عناصر ، يحتوي على خصائص اضافية مثل:

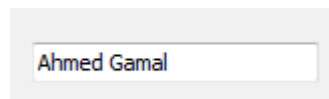
CheckState: حالة الاختيار.

ThreState: لتحديد كونه يحمل خاصية ثالثة (مفعل - غير مفعل - مفعل جزئياً) .

6. 3. RadioButton

مثل السابق ولكنه يسمح لك باختيار واحد، لو كنت تريد استخدامه لأكثر من مرة ضع كل منهم في panel مختلف.

6 .4. ComboBox و ListBox

قائمة للاختيار من ضمن خيارات ، الاولى في صف واحد والثانية حسب الحجم المطلوب.

يحتوي على جميع الخصائص التي تعلمناها في Collection.List مثل Items.Add و Items.Remove ....

6 .5. TextBox

أداة نصية تسمح للمستخدم بالكتابة داخلها ، بعض خصائصها الاضافية :

ReadOnly : جعلها للقراءة فقط.

PasswordChar : الرمز الذي يظهر في حالة جعل مربع النص للادخال.

MultiLine : السماح بجعلها متعددة الأسطر.

MaxLength : الطول الاقصى للنص.

ScrollsBar : اشرطة التمرير.

WordWrap : التفاف النص بعد وصوله إلى حافة الأداة مثل برنامج وورد مثلاً .

حيث انها مشتقة من الفئة System.Windows.Forms.ScrollableControl، لذا سيكون بامكاننا اختيار وضع من ال enumeration المسمى ScrollBars.

تطبيقات سريعة على مربع النصوص

- لجعل مربع النصوص لا يقبل سوى ارقام مثلاً :

كود C#

```
private void TextBox1_KeyPress(object sender,
System.Windows.Forms.KeyPressEventArgs e)
{
    if ((e.KeyChar < '0' | e.KeyChar > '9'))
    {
        e.Handled = true;
    }
}
```

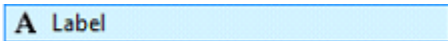
كود VB.NET

```
Private Sub TextBox1_KeyPress(ByVal sender As Object, ByVal e As
System.Windows.Forms.KeyPressEventArgs) Handles TextBox1.KeyPress
    If (e.KeyChar < "0"c Or e.KeyChar > "9"c) Then
        e.Handled = True
    End If
End Sub
```

- البحث عن كلمة داخل مربع النص :

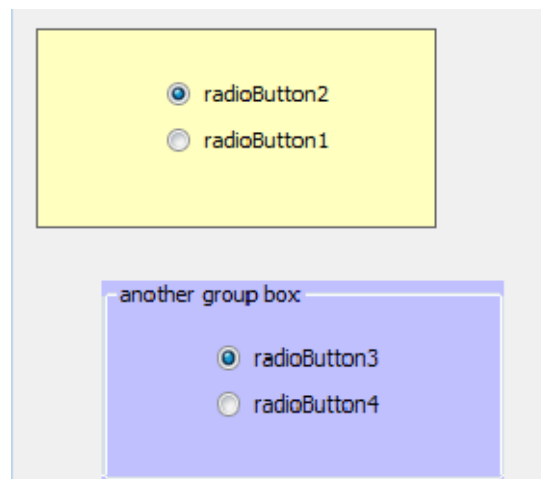
كود C#

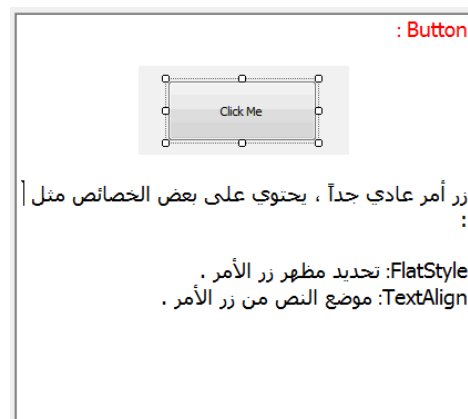
```
private void Button1_Click(object sender, System.EventArgs e)
{
    // البحث نتيجة مكان اول على سيحتوي الذي المتغير
    int index;
    //البحث كلمة
    string SearchWord = "Ahmed";

    //للأداة الانتقال
    TextBox1.Focus();
    //البداية في عرفناه الذي المتغير في الكلمة بداية مكان وضع
    index = TextBox1.Text.IndexOf(SearchWord);
    // وبطولها الكلمة بداية من التحديد خصائص باستخدام الكلمة تحديد
    TextBox1.SelectionStart = index;
    TextBox1.SelectionLength = SearchWord.Length;
}
```

كود	VB.NET

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
    ' البحث نتيجة مكان اول على سيحتوي الذي المتغير
    Dim index As Integer
    'البحث كلمة
    Dim SearchWord As String = "Ahmed"

    'للأداة الانتقال
    TextBox1.Focus()
    'البداية في عرفناه الذي المتغير في الكلمة بداية مكان وضع
    index = TextBox1.Text.IndexOf(SearchWord)
    ' وبطولها الكلمة بداية من التحديد خصائص باستخدام الكلمة تحديد
    TextBox1.SelectionStart = index
    TextBox1.SelectionLength = SearchWord.Length
End Sub
```

6 .6. Label

أداة لعرض النصوص غير قابلة للتعديل – فقط .

6 .7. GroupoBox و Panel

أدوات حاوية يمكن وضع أدوات أخرى بداخلها.

6. 8. RichTextBox

أداة نص تسمح لمحتوياتها ان تحتوي على تنسيقات مختلفة ، الفارق بينها وبين TextBox هو الفارق بين Notepad و Wordpad،

تحتوي على الخصائص الاضافية التالية:

Wordwrap: لتحديد التفاف النص.

Lines: لتحديد عدد الأسطر .

Select: لتحديد جزء من النص عن طريق نقطة بداية ونهاية ، او نقطة بداية وطول.

SelectionXXX: لتحديد تنسيق خط للجزء المحدد ، مثل اللون SelectionColor.

DetectUrls : لتحديد عناوين الانترنت تلقائياً وجعلها قابلة للانتقال .

كما يحتوي على الدوال الاضافية التالية:

LoadFile(): لفتح ملف نصي من مسار معين.

SaveFile(): لحفظ الملف في مكان معين.

أمثلة سريعة

- لقراءة محتويات اداة النص من ملف نصي ومن ثم الحفظ في ملف آخر :

C# كود
```
RichTextBox1.LoadFile(@"C:\Ahmed.txt", RichTextBoxStreamType.PlainText);
RichTextBox1.SaveFile(@"C:\Ahmed2.txt", RichTextBoxStreamType.PlainText);
```

VB.NET كود
```
RichTextBox1.LoadFile("C:\Ahmed.txt", RichTextBoxStreamType.PlainText)
RichTextBox1.SaveFile("C:\Ahmed2.txt", RichTextBoxStreamType.PlainText)
```

ولو كان الملف نصي يحتوي على تنسيقات من النوع rtf فسنقوم بضبط النوع إلى RichTextBoxStreamType.RichText بالشكل التالي مثلاً :

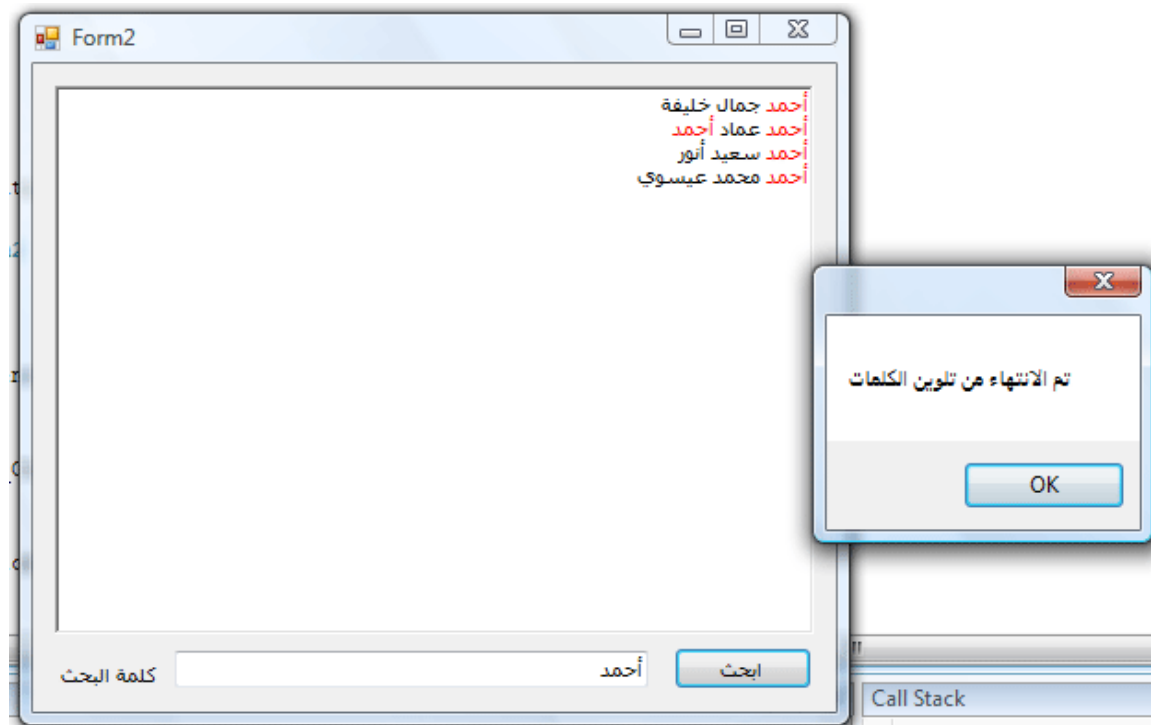
C# كود
```
RichTextBox1.LoadFile(@"C:\Ahmed.rtf ", RichTextBoxStreamType.RichText);
RichTextBox1.SaveFile(@"C:\Ahmed2.rtf ", RichTextBoxStreamType.RichText);
```

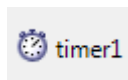
VB.NET كود
```
RichTextBox1.LoadFile("C:\Ahmed.rtf ", RichTextBoxStreamType.RichText)
RichTextBox1.SaveFile("C:\Ahmed2.rtf", RichTextBoxStreamType.RichText)
```

والآن سنعيد تطبيق مثال البحث الذي سبق عمله لأداة النص ، ولكن هذه المرة سنقوم بتلوين كل نتائج البحث بدلاً من تحديدها واحدة بالأخرى :

C# كود
```
int index = 0;
while ((index = richTextBox1.Text.IndexOf(textBox1.Text, index)) != -1)
{

    richTextBox1.Select(index, textBox1.Text.Length);
    richTextBox1.SelectionColor = System.Drawing.Color.Red;

    index += textBox1.Text.Length;

}
MessageBox.Show("تم الانتهاء من تلوين الكلمات");
```

VB.NET كود
```
Dim index As Integer = 0
While (index = richTextBox1.Text.IndexOf(textBox1.Text, index)) <> -1
    richTextBox1.[Select](index, textBox1.Text.Length)
    richTextBox1.SelectionColor = System.Drawing.Color.Red
    index += textBox1.Text.Length
End While
```

وسيكون ناتج التنفيذ بالشكل التالي :

الصورة 14. 13. نتائج تنفيذ شفرة البحث و التلوين.

6. 9. Timer

أداة للمؤقت ، لها حدث Ticker والذي يتم تنفيذه كل مدة معينة Interval ، تفيد في حالة عمل اوامر تكراريه بفوارق زمنية.

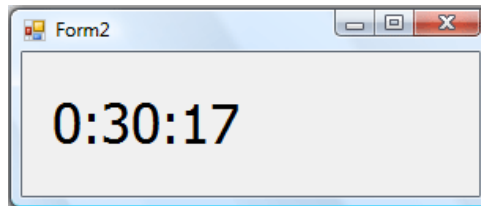
مثال عمل ساعة بالثواني ، قم بضبط خاصية Interval إلى 1000 وهو ما تعادل ثانية واحدة ، ايضاً قم بضبط الخاصية Enabled إلى True ايضاً وقم بكتابة الكود التالي في الحدث Tick:

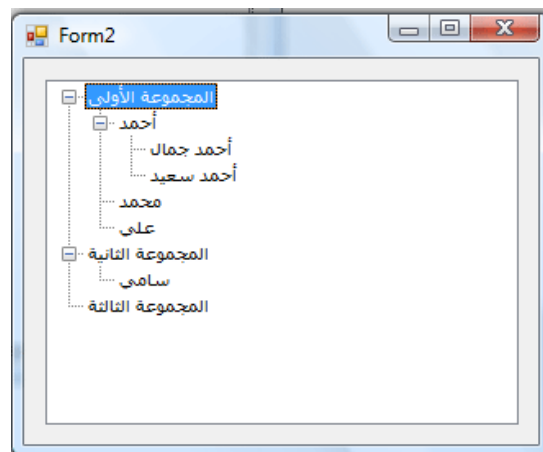
كود	C#

```
label1.Text = DateTime.Now.Hour + ":" + DateTime.Now.Minute + ":" +
DateTime.Now.Second;
```

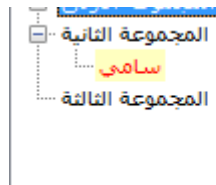
كود	VB.NET

```
label1.Text = DateTime.Now.Hour + ":" + DateTime.Now.Minute + ":" +
DateTime.Now.Second
```

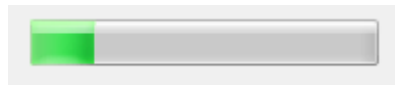
وسيكون الناتج شيئاً مثل هذا :

الصورة 14. 14. نتائج تنفيذ الشفرة.

6. 10. TreeView

الصورة 14. 15.الكائن TreeView.

لعرض أداة الشجرة ، عنصرها الاساسي هو Nodes والذي يمكن من خلاله اضافة أي عناصر وحذفها.

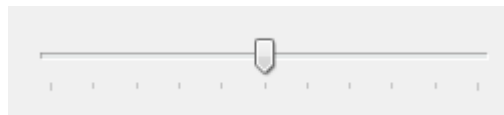
يمكن التغيير في التنسيق وحتى اضافة صور وخلافه ضمن هذه الأداة :

يمكنك التعرف على المزيد عنها من الرابط التالي :

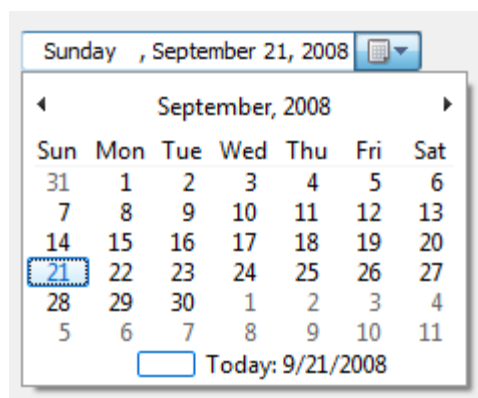
رابط

http://msdn.microsoft.com/en-us/library/system.windows.forms.treeview.aspx

6. 11. ProgressBar

لعرض مؤشر التقدم ، خصائصه الاساسية هي Minimum - Maximum لتحديد الحد الأعلى والأدنى اضافة للخاصية Step لتحديد مقدار التقدم كل مرة ، يمكنك تحديد Style لعملية التقدم من خلال الخاصية Style والتي تقرأ من ال enumeration المسمى ProgressBarStyle.

يمكن تحديد الخاصية لاحقاً باستخدام الخاصية Value.

6. 12. TrackBar

مشابه لل Progressbar لكنك انت من تتحكم بقيمته ، مثل Trackbar الخاص برفع وخفض الصوت.

5 .13. DateTimePicker

الصورة 14 .16.الكائن DateTimePicker.

أداة لاختيار التواريخ والأوقات ، يمكنك تحديد اسلوب العرض من الخاصية Format لتحديد اسلوب التاريخ المعروض . يمكن الحصول على التاريخ المعروض من الخاصية Value كما يمكن وضع حد اقصى للتاريخ وحد أدنى عن طريق الخاصيتين MaxDate - MinDate .

الأدوات كثيرة جداً ولا جدوى من اضاعة الوقت في التعرف على تفاصيلها ... لذا أكتفي بأن احيلك إلى موقع مايكروسوفت حيث تجد شرحاً تفصيلياً لكل أداة :

رابط

http://msdn.microsoft.com/en-us/library/aa984065(VS.71).aspx

7. Dialogs

في بداية موضوعنا عن المربعات الحوارية سنقوم أولاً بتصميم واحد خاص بنا ، ثم في المرحلة الثانية سنقوم بالتعرف على استخدامات بعض المربعات الحوارية الشائعة.

قم بانشاء فورم جديد ، قم بالغاء خاصية ال Resizeable ، قم بضبط خاصية FormBorderStyle إلى FixedDialog ، قم باخفاء MinimizeBox و MaximizeBox ... بهذه الطريقة يكون تصميم أغلب المربعات الحوارية.

الآن قم بتصميم المربع الحواري كما يحلو لك ، في اي مربع حواري أو Dialog يكون الناتج واحداً من العناصر المعرفة في enumeration المسمى DialogResult والذي يحتوي على القيم التالية:

Abort, Cancel, Ignore, No, None, OK, Retry, Yes

لضبط الزر الذي يعيد القيمة OK نضبط خاصية ال AcceptButton للفورم على الزر المختار.

ايضاً الامر بسيط بالنسبة للخاصية Cancel حيث نجد الخاصية CancelButton لاظهار فورم بصورة مربع حواري Dialog نستخدم الأمر ShowDialog بالشكل التالي مثلاً:

كود C#

```
newDialog.ShowDialog();
```

كود VB.NET

```
newDialog.ShowDialog()
```

ولكن وبما اننا بحاجة لقراءة القيم الناتجة عنه ومعرفة اي زر تم اختياره ، سنكتب الكود الخاص بنا بالشكل التالي على سبيل المثال:

كود C#

```
if (newDialog.ShowDialog() == DialogResult.Cancel)
{
    // do something
}
else if (newDialog.ShowDialog() == DialogResult.Ok)
{
    // do something else
}
```

كود VB.NET

```
If newDialog.ShowDialog() = DialogResult.Cancel Then
    ' do something
ElseIf newDialog.ShowDialog() = DialogResult.OK Then
    ' do something else
End If
```

ولو حاولنا قراءة محتويات ال Dialog سيكون لزاماً علينا تعريف القيم المطلوبة public ، لذا قم مثلاً بتعديل مربع النص بالشكل التالي:

كود C#

```
public System.Windows.Forms.TextBox TextBox1;
```

كود VB.NET

```
Public TextBox1 As System.Windows.Forms.TextBox
```

7. 1. MessageBox

النوع الابسط والأسهل من ال Dialogs هو رسائل التحذير MessageBox ، صورتها الأبسط بالشكل التالي:

كود C#

```
MessageBox.Show("Welcome ...");
```

كود VB.NET

```
Public TextBox1 As System.Windows.Forms.TextBox
```

لو جربت معرفة البارميترات التي يمكن ارسالها للدالة Show ستجد 20 overloads، تتيح لك التحكم بأي شكل في رسائلك التحذيرية ، سنستعرض اثنين منها هنا:

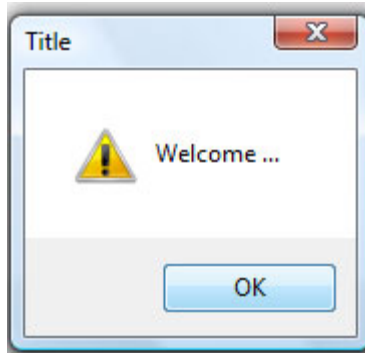
- اظهار رسالة تحذيرية كاملة

كود C#

```
MessageBox.Show("Welcome ...", "Title", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);
```

كود VB.NET

```
MessageBox.Show("Welcome ...", "Title", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation)
```

المظهر:

- اظهار رسالة تحذيرية تخيرك بين عدة حالات ، مع جعل واحد منها افتراضياً

كود C#

```
DialogResult reslt = MessageBox.Show("Do you want to save", "Title",
MessageBoxButtons.YesNoCancel, MessageBoxIcon.Exclamation,
MessageBoxDefaultButton.Button1);
if (reslt = DialogResult.Yes)
    MessageBox.Show("you want to save.");
else if (reslt = DialogResult.No)
    MessageBox.Show("you don't want to save.");
else if (reslt = DialogResult.Cancel)
    MessageBox.Show("you cancel this operation.");
```

كود VB.NET

```
Dim reslt As DialogResult = MessageBox.Show("Do you want to save", "Title",
MessageBoxButtons.YesNoCancel, MessageBoxIcon.Exclamation,
MessageBoxDefaultButton.Button1)
If reslt = DialogResult.Yes Then
    MessageBox.Show("you want to save.")
ElseIf reslt = DialogResult.No Then
    MessageBox.Show("you don't want to save.")
ElseIf reslt = DialogResult.Cancel Then
    MessageBox.Show("you cancel this operation.")
End If
```

هناك عدة اختيارات أخرى ، تتيح لك اظهار ملف مساعدة ... الخ ، يمكنك استعراضها جميعاً من خلال هذا الرابط:

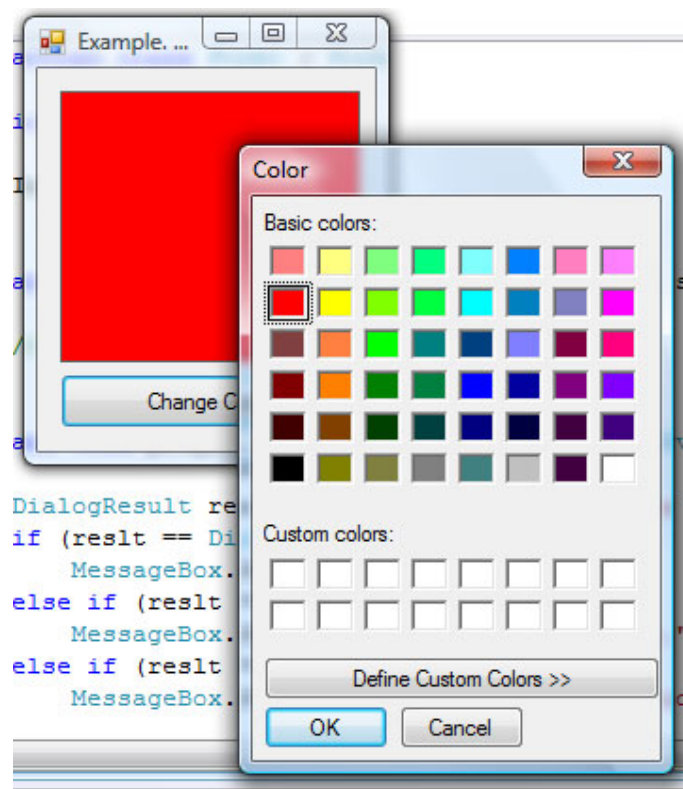
رابط

http://msdn.microsoft.com/en-us/library/system.windows.messagebox.show.aspx

7 .2. Dialog Controls

بعد ان تعرفنا على النوع الأبسط من المربعات الحوارية وتعلمنا كيفية انشاءها ، جاء الدور الآن على تعلم كيفية استخدام المربعات الحوارية الأساسية الموجودة ضمن الأدوات...

7 .3. ColorDialog

مربع اختيار الألوان ، قم باضافته من ال ToolBox وسننشأ الآن مثالاً بسيطاً لتغيير لون الخلفية لصورة مثلاً:

الصورة 14 .17.الكائن ColorDialog.

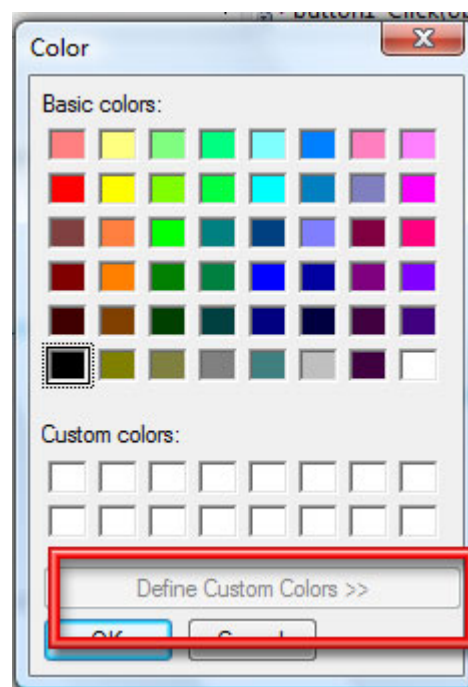
ضف صورة PictureBox زر أمر بسيط واكتب فيه الأمر التالي:

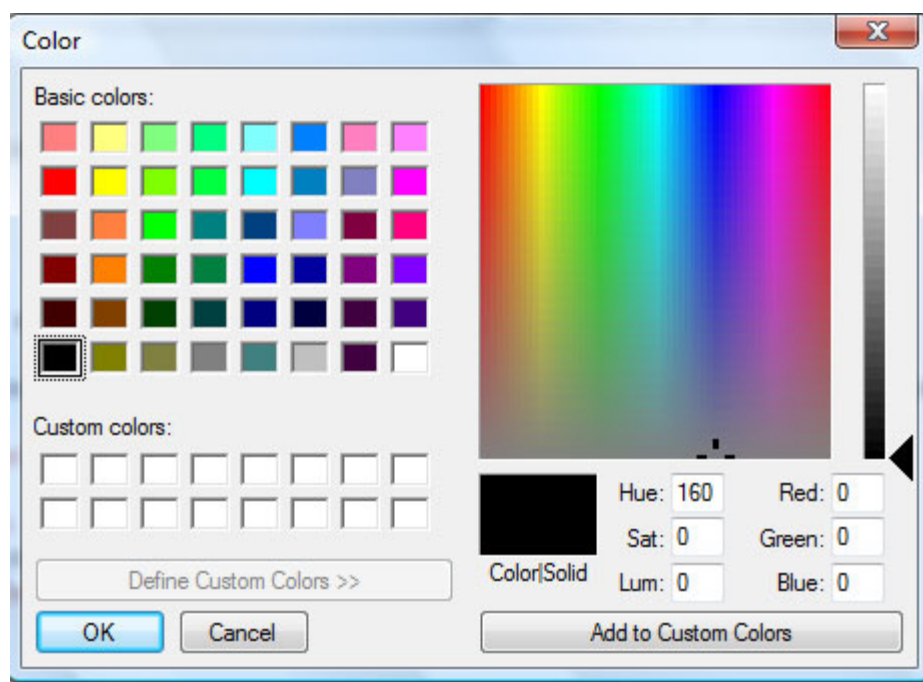
كود	C#

```
if (colorDialog1.ShowDialog() == DialogResult.OK)
    pictureBox1.BackColor = colorDialog1.Color;
else
    MessageBox.Show("cancel operation");
```

كود	VB.NET

```
If colorDialog1.ShowDialog() = DialogResult.OK Then
    pictureBox1.BackColor = colorDialog1.Color
Else
    MessageBox.Show("cancel operation")
End If
```

هناك بعض الخصائص الاضافية في مربع الحوار هذا لكنها غير شائعة الاستخدام مثل AllowFullOpen والتي تستطيع منع المستخدم من اختيار غير الالوان القياسية بالشكل التالي مثلاً:

الصورة 14. 18.الكائن ColorDialog مع الاتسغناء عن لوحة اختيار الألوان المتقدمة.

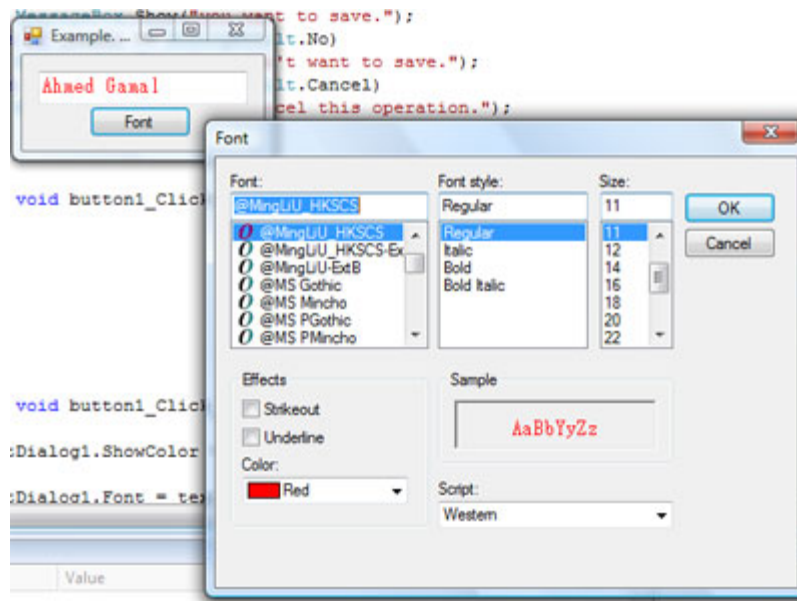
والخاصية FullOpen التي تظهرها بالكامل مباشرة بالشكل التالي مثلاً:

الصورة 14. 19. الكائن ColorDialog مع لوحة اختيار الألوان المتقدمة.

رابط تفصيلي من مايكروسوفت:

رابط

http://msdn.microsoft.com/en-us/library/ms646375.aspx

7. 4. FontDialog

يستخدم لاختيار الخطوط، سنقوم الآن بعمل مثال لتغيير خط مربع نص:

الصورة 14. 20.الكائن FontDialog .

كود	C#

```
fontDialog1.ShowColor = true;
fontDialog1.Font = textBox1.Font;
fontDialog1.Color = textBox1.ForeColor;
if (fontDialog1.ShowDialog() == DialogResult.OK)
{
    textBox1.Font = fontDialog1.Font;
    textBox1.ForeColor = fontDialog1.Color;
}
```

كود	VB.NET

```
fontDialog1.ShowColor = True
fontDialog1.Font = textBox1.Font
fontDialog1.Color = textBox1.ForeColor
If fontDialog1.ShowDialog() = DialogResult.OK Then
    textBox1.Font = fontDialog1.Font
    textBox1.ForeColor = fontDialog1.Color
End If
```

رابط تفصيلي من مايكروسوفت:

رابط

http://msdn.microsoft.com/en-us/library/system.windows.forms.fontdialog(VS.80).aspx

7 .5. Open And Save Dialogs

اشهر نوعين من المربعات الحوارية ، في العادة يتم استخدامهم لتحديد عملية فتح الملفات أو تحديد مكان حفظها ، سنقوم بعمل مثال بسيط نقوم من خلال بفتح ملف نصي ، التعديل عليه ومن ثم حفظه:

اضف مربع نص بسيط واجعل خاصية Multiline=True ، ضف زر للحفظ وآخر للفتح ، وأخيراَ لن نقوم باضافة OpenFileDialog و SaveFileDialog بل سنضيفهم من خلال الكود مباشرة.

أول عملية لدينا هي استخدام المربع الحواري لتحديد الملفات النصية ومن ثم اختيار واحد منها:

كود C#

```
string fileToOpen = "";
OpenFileDialog openFile = new OpenFileDialog();
openFile.DefaultExt = "txt";
openFile.Filter = "Word documents (*.txt)|*.txt";
```

كود VB.NET

```
Dim fileToOpen As String = ""
Dim openFile As New OpenFileDialog()
openFile.DefaultExt = "txt"
openFile.Filter = "Word documents (*.txt)|*.txt"
If openFile.ShowDialog() = DialogResult.OK Then
    fileToOpen = openFile.FileName
End If
```

الآن اصبح لدينا مسار الملف في متغير، سنقوم بعملية وضع محتوياته في مربع نص بطريقة تعلمناها سابقاً:

كود C#

```
if (fileToOpen != "")
{
    using (System.IO.StreamReader s1 = System.IO.File.OpenText(fileToOpen))
    {
        string input = null;
        textBox1.Text = "";
        while ((input = s1.ReadLine()) != null)
        {
            textBox1.Text += input;
        }

    }
}
```

كود	VB.NET

```
If fileToOpen <> "" Then
    Using s1 As System.IO.StreamReader = System.IO.File.OpenText(fileToOpen)
        Dim input As String = Nothing
        textBox1.Text = ""
        While (input = s1.ReadLine()) IsNot Nothing
            textBox1.Text += input


        End While
    End Using
End If
```

الآن سنستخدم مربع حوار الحفظ لتحديد المكان الذي نود فيه حفظ الملف:

كود	C#

```
string fileToSave = "";
SaveFileDialog saveFileDialog1 = new SaveFileDialog();
saveFileDialog1.Filter = "txt files (*.txt)|*.txt|All files (*.*)|*.*";
if (saveFileDialog1.ShowDialog() == DialogResult.OK)
    fileToSave = saveFileDialog1.FileName;
```

كود	VB.NET

```
Dim fileToSave As String = ""
Dim saveFileDialog1 As New SaveFileDialog()
saveFileDialog1.Filter = "txt files (*.txt)|*.txt|All files (*.*)|*.*"
If saveFileDialog1.ShowDialog() = DialogResult.OK Then
    fileToSave = saveFileDialog1.FileName
End If
```

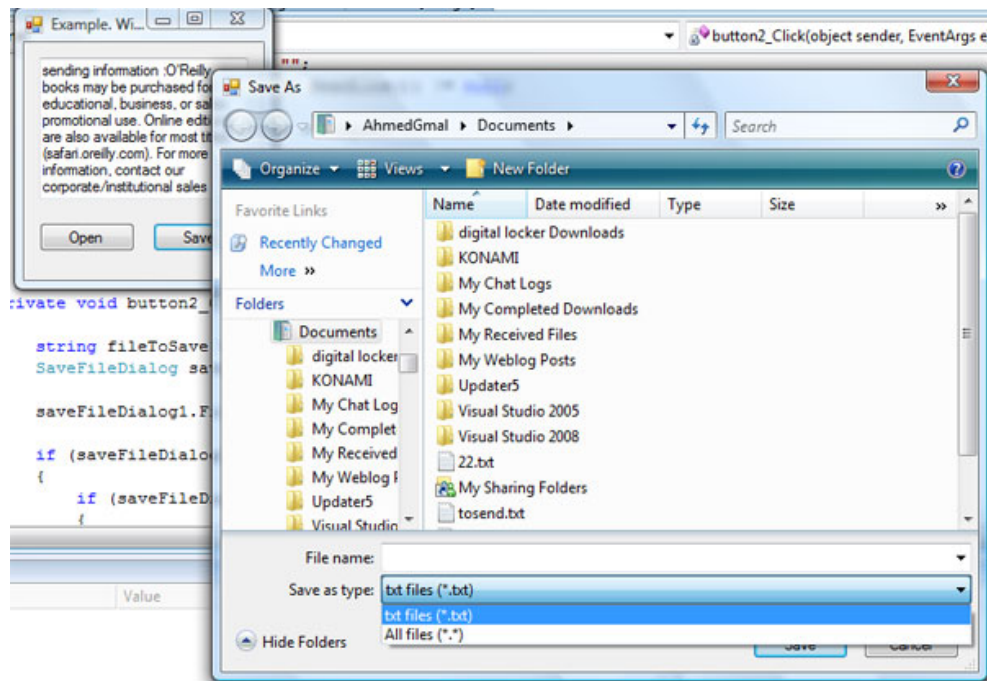
وكما تعلمنا سابقاً ، نقوم بحفظ الملف النصي في المسار الذي قام المستخدم باختياره:

كود	C#

```
if (fileToSave != "")
{
    using (System.IO.StreamWriter w1 = System.IO.File.CreateText(fileToSave))
    {
        w1.WriteLine(textBox1.Text);
    }
}
```

كود	VB.NET

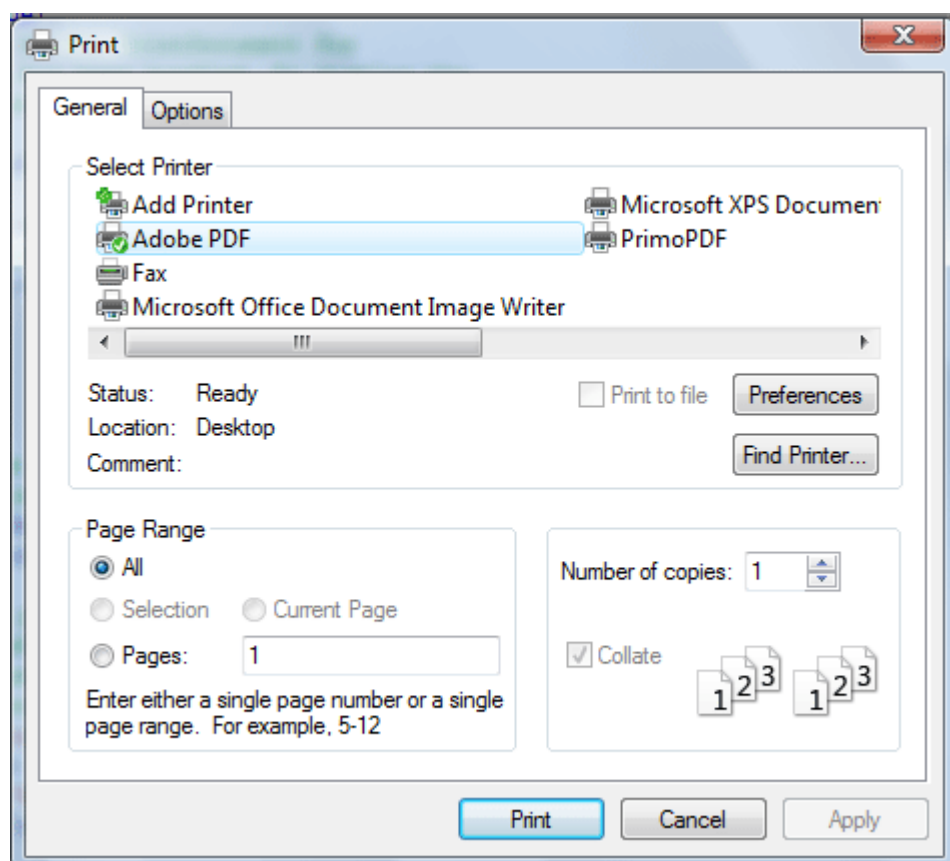
```
If fileToSave <> "" Then
    Using w1 As System.IO.StreamWriter = System.IO.File.CreateText(fileToSave)
        w1.WriteLine(textBox1.Text)

    End Using
End If
```

الناتج سيكون شيئاً مثل هذا :

الصورة 14. 21.الكائن OpenFileDialog.

هناك بعض الخصائص الاضافية لكلا الكائنين ، يمكنك الاطلاع على المزيد عنهما هنا:

 رابط

http://msdn.microsoft.com/en-us/library/aa287592(VS.71).aspx
http://msdn.microsoft.com/en-us/library/system.windows.forms.savefiledialog.aspx

حيث يمكنك تحديد نوع من الملفات او قراءة اسم المجلد او اسم الملف فقط ، تحديد أكثر من ملف للفتح ... الخ.

PrintDialog .6 .6

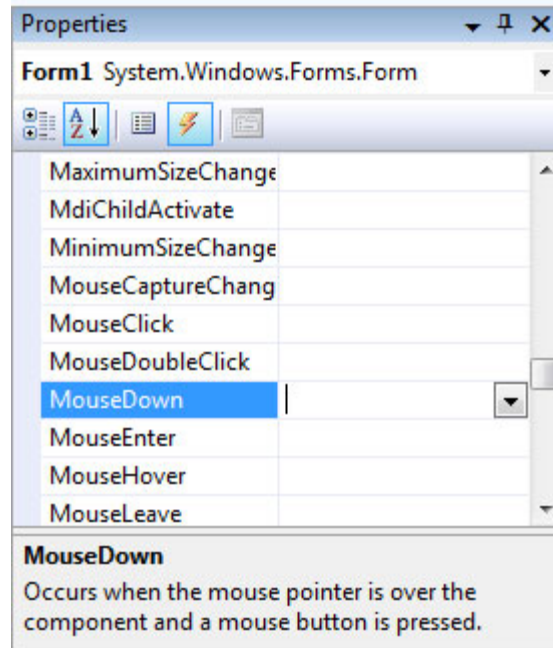
الصورة 14. 22.الكائن PrintDialog.

يستخدم للتحكم في خصائص عملية الطباعة ، يمكن ان يفيدك الرابط التالي من مايكروسوفت:

رابط

http://msdn.microsoft.com/en-us/library/system.windows.forms.printdialog.aspx

8. أحداث الماوس

أولاً لمعرفة الزر المضغوط من الماوس، سنذهب إلى الحدث MouseDown من نافذة ال Properties

الصورة 14. 23. تحديد اسم الدالة التي تنفذ عند وقوع الحدث MouseDown من نافذة الخصائص

ومن ثم نكتب الكود التالي:

C#	كود

```
if (e.Button == MouseButtons.Left)
    MessageBox.Show("Left click");
if (e.Button == MouseButtons.Right)
    MessageBox.Show("Right click");
if (e.Button == MouseButtons.Middle)
    MessageBox.Show("Middle click");
```

VB.NET	كود

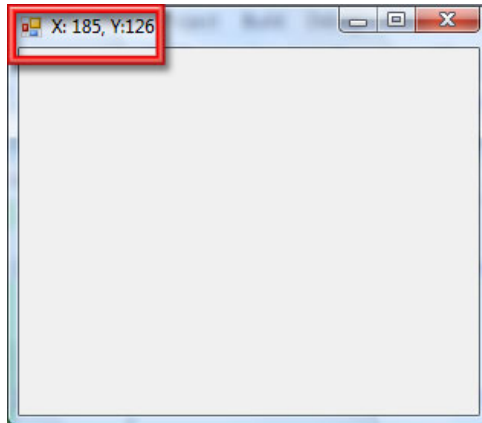
```
If e.Button = MouseButtons.Left Then
    MessageBox.Show("Left click")
End If
If e.Button = MouseButtons.Right Then
    MessageBox.Show("Right click")
End If
If e.Button = MouseButtons.Middle Then
    MessageBox.Show("Middle click")
End If
```

ولمعرفة النقطة التي يمر عليها الماوس مثلاً في حدث MouseMove :

كود C#

```
this.Text = "X: " + e.X + ", Y:" + e.Y;
```

كود VB.NET

```
Me.Text = "X: " + e.X + ", Y:" + e.Y
```

9. أحداث الكيبورد

لقراءة الحدث الخاص بالزر المضغوط نكتب الأمر التالي مثلاً في حدث: KeyUp

كود C#

```
this.Text = "key: " + e.KeyCode.ToString();
```

كود VB.NET

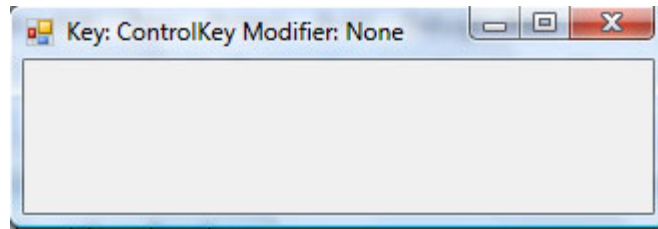
```
Me.Text = "key: " + e.KeyCode.ToString()
```

ولكن لمعرفة ما إذا كان هناك زر آخر مضغوط مثلاً مع الزر الحالي، سنضع الزر الأول في key والثاني في Modifier بالشكل التالي:

كود C#

```
this.Text = string.Format("Key: {0} Modifier: {1}", e.KeyCode.ToString(),
e.Modifiers.ToString());
```

كود	VB.NET

```
Me.Text = String.Format("Key: {0} Modifier: {1}", e.KeyCode.ToString(),
e.Modifiers.ToString())
```

طبعاً لا داعي لاخبارك ان بامكانك قراءة الأحداث باستخدام If وتنفيذ بعض العمليات بناء عليها

الباب

15

GDI+

.1 مقدمة إلى GDI+

تتبع هذه الأوامر مجال الأسماء System.Drawing وتوفر لك حلولاً افضل من اجل رسومات D2 على الفورم والمخططات البيانية وخلافه ، تحتوي على الفئات الرئيسية التالية:

1. System.Drawing: الفئة الرئيسية ، تحتوي على الأقلام والفرش الاساسية التي تستخدمها في عمليات الرسم.
2. System.Drawing.Drawing2D : تقدم مجموعة أخرى من الدوال التي تساعدك على الرسم مثل gradient brushes و geometric transforms
3. System.Drawing.Imaging : تساعدك في عملياتك على الصور وقراءة البيانات الداخلية وتنفيذ العمليات المختلفة.
4. System.Drawing.Printing : تساعدك على تحويل رسومات إلى صور للطباعة والتعامل مع الطاعبة.
5. System.Drawing.Text: التعامل بصورة رسومية مع الخطوط وخلافه.

.2 محتويات مجال الأسماء System.Drawing

هو مجال الاسماء الاساسي والأكثر استخداماً ،أهم محتوياته هي:

العنصر	الوصف
Bitmap	لاحتواء معلومات الصورة
Brush	فرشاة لعملية التلوين ، ولها عدة انواع
BufferedGraphics	يمكنك باستخدام هذا ال Buffer الرسم باستخدام تقنية double buffering
Color	SystemColors الألوان التي يمكن استخدامها في عمليات الرسم والتلوين
Font	FontFamily الخطوط التي يمكن استخدامها في عمليات رسم النصوص.
Graphics	الكائن الاساسي في عمليات الرسم ، والتي سنتعرف عليها لاحقاً

Icon	SystemIcons التعامل مع الايقونات
Image	ال abstract class لكل ما يتعلق بعمليات الصور
ImageAnimator	للتعامل مع الصور المتحركة
Pen	قلم لعمليات الرسم ، وله عدة أنواع
Point	للتعامل مع الاحداثيات
Rectangle	مستطيل
Size	حساب المساحة للشكل
StringFormat	تحتوي هذه الفئة على كل ما يتعلق بعمليات النصوص
Region	خاص بالتعامل مع الاشكال أي كانت ، وله عدة خصائص مفيدة جداً

الجدول 15. 1. أهم فئات مجال الأسماء System.Drawing.

Double Buffering

هي تقنية تسمح للرسم بالاكتمال في buffer مستقل ثم ظهوره للمستخدم مرة واحدة لتلافي مشكلة flicker، على العموم هي تقنية مشهورة في تقنيات الرسم يمكن التعرف عليها من هنا:

رابط

http://msdn.microsoft.com/en-us/library/b367a457.aspx

كما يمكنك معرفة المزيد عن جميع محتوياتها من خلال هذا الرابط:

رابط

http://msdn.microsoft.com/en-us/library/system.drawing.aspx

3. الفئة Graphics

الكائن Graphics هو الكائن الأساسي في عملية الرسم ، يمكن انشاءه بعدة طرق:

- من الفورم ، وينطبق عليه من PictureBox وخلافه:

كود C#

```
Graphics myGraphic = Form1.CreateGraphics();
```

كود VB.NET

```
Dim myGraphic As Graphics = Form1.CreateGraphics()
```

- أو من حدث الرسم لأي اداة كالفوم مثلاً ، عن طريق الكائن PaintEventArgs بالشكل التالي مثلاً:

كود C#

```
private void Form1_Paint(object sender, PaintEventArgs e)
{
    Graphics myGraphic = e.Graphics();
}
```

كود VB.NET

```
Private Sub Form1_Paint(ByVal sender As Object, ByVal e As PaintEventArgs)
    Dim myGraphic As Graphics = e.Graphics()
End Sub
```

ويحتوي هذا الكائن على الأوامر التالية:

العنصر	الوصف
FromHdc()	عمل كائن رسم من صورة موجودة حالياً
FromHwnd()	
FromImage()	
Clear()	مسح محتويات الكائن
DrawArc()	رسم صورة او شكل هندسي
DrawBeziers()	رسم - Beziers لا أعرف معناها ولكنها ترسم منحنيات. -
DrawCurve()	رسم منحنى
DrawEllipse()	رسم شكل بيضاوي
DrawIcon()	رسم ايقونة

DrawLine()	رسم خط مستقيم
DrawLines()	رسم مجموعة من الخطوط
DrawPie()	رسم مخطط بياني
DrawPath()	رسم مسار
DrawRectangle()	رسم مستطيل
DrawRectangles()	رسم مستطيلات
DrawString()	رسم نص
FillEllipse()	تلوين مضلع
FillPie()	تلوين مخطط بياني
FillPolygon()	تلوين شكل بيضاوي
FillRectangle()	تلوين مربع
FillPath()	تلوين مسار

الجدول 15. 2. أهم دوال الفئة Graphics

لمزيد من المعلومات حول هذه الفئة ودوالها:

رابط

http://msdn.microsoft.com/en-us/library/system.drawing.graphics_members.aspx

والآن لعلك لاحظت أن جميع دوال الرسم DrawXXX تستخدم القلم Pen، أما دوال التلوين FillXXX فهي تستخدم الفرشاة Brush، لذا سنبدأ بالتعرف على هذين الكائنين أولاً.

4. الفئة Pen

يمكن استخدام Pen في عمليات الرسم مباشرة عن طريق تحديد مثل Pens.Blue للقلم الأزرق وخلافه ، إلا أننا ما زلنا قادرين على تعريف كائن منه والاستفادة من خصائصه المتعددة بالشكل التالي مثلاً:

كود C#
```
Pen myPen = new Pen(Color.Black, 3);
```

كود VB.NET
```
Dim myPen As New Pen(Color.Black, 3)
```

حيث قمنا بانشاء قلم بلون أسود وبعرض 3. يمكننا استخدام المزيد من الخصائص للقلم عن طريق تحديد مثلاً شكل نقطة البداية:

كود C#
```
myPen.StartCap = LineCap.ArrowAnchor;
```

كود VB.NET
```
myPen.StartCap = LineCap.ArrowAnchor
```

لمعرفة المزيد عن خصائص القلم:

رابط

http://msdn.microsoft.com/en-us/library/system.drawing.pen.aspx

5. الفئة Brush

بنفس الطريقة ، يمكن انشاء فرشاة باللون الأصفر كأبسط مثال بالشكل التالي:

كود C#
```
SolidBrush myBrush = new SolidBrush(Color.Yellow);
```

كود VB.NET
```
Dim myBrush As New SolidBrush(Color.Yellow)
```

يمكن ايضاً إنشاء فرشاة بأكثر من لون بالشكل التالي مثلاً:

كود C#

```
HatchBrush myBrush = new HatchBrush(HatchStyle.BackwardDiagonal, Color.Green,
Color.White);
```

كود VB.NET

```
Dim myBrush As New HatchBrush(HatchStyle.BackwardDiagonal, Color.Green,
Color.White)
```

لمعرفة المزيد عن خصائص الفرشاة:

رابط

http://msdn.microsoft.com/en-us/library/system.drawing.brush.aspx

6. الرسم

بعد ان انشأنا كائننا الخاص للرسم ، يمكننا البدء في رسم خط مستقيم بالشكل التالي مثلاً:

كود C#

```
myGraphic.DrawLine(Pens.Blue, 20, 20, 100, 100);
```

كود VB.NET

```
myGraphic.DrawLine(Pens.Blue, 20, 20, 100, 100)
```

أو مستطيل:

كود C#

```
myGraphic.DrawRectangle(Pens.Blue, New Rectangle(20, 20, 100, 100));
```

كود VB.NET

```
myGraphic.DrawRectangle(Pens.Blue, New Rectangle(20, 20, 100, 100))
```

أو رسم منحنى:

كود C#

```
myGraphic.DrawBezier(Pens.Blue, 10, 20, 50, 80, 10, 80, 100, 50);
```

كود VB.NET

```
myGraphic.DrawBezier(Pens.Blue, 10, 20, 50, 80, 10, 80, 100, 50)
```

بامكاننا رسم خط مستقيم ولكن باستخدام كائن قلم سبق تعريفه وتحديده بأنه منقط بالشكل التالي مثلاً:

كود C#

```
Graphics myGraphic = e.Graphics;
Pen myPen = new Pen(Color.Blue, 3);
myPen.DashStyle = System.Drawing.Drawing2D.DashStyle.Dot;
myGraphic.DrawLine(myPen, 20, 20, 100, 100);
```

كود VB.NET

```
Dim myGraphic As Graphics = e.Graphics
Dim myPen As New Pen(Color.Blue, 3)
myPen.DashStyle = System.Drawing.Drawing2D.DashStyle.Dot
myGraphic.DrawLine(myPen, 20, 20, 100, 100)
```

أو لرسم مسار من عدة خطوط على شكل مثلث مثلاً:

كود C#

```
GraphicsPath myGraphicPath = new GraphicsPath();
myGraphicPath.StartFigure();
myPath.AddLine(10, 10, 30, 60);
myPath.AddLine(30, 60, 60, 10);
myPath.AddLine(60, 10, 10, 10);
myGraphicPath.CloseFigure();
myGraphic.DrawPath(myPen, myGraphicPath);
```

كود VB.NET

```
Dim myGraphicPath As New GraphicsPath()
myGraphicPath.StartFigure()
myPath.AddLine(10, 10, 30, 60)
myPath.AddLine(30, 60, 60, 10)
myPath.AddLine(60, 10, 10, 10)
myGraphicPath.CloseFigure()
myGraphic.DrawPath(myPen, myGraphicPath)
```

أو لرسمه ملوناً نغير السطر الأخير ليكون FillPath مع تمرير الفرشاة المناسبة

7. رسم النصوص

رسم النصوص هو قسم آخر من عالم ال GDI أبسط مثال عليه هو الكود التالي لرسم نص بفرشاة حمراء وبلون أحمر في النقطة 200 و 200.

كود C#

```
myGraphic.DrawString("Hello GDI+", new Font("Times New Roman", 30),
Brushes.Red, 200, 200);
```

كود VB.NET

```
myGraphic.DrawString("Hello GDI+", new Font("Times New Roman", 30),
Brushes.Red, 200, 200)
```

لكن لاحقاً يكون بامكاننا استخدام الكائن StringFormat واضافته إلى متغيرات الرسم ، لكي نرسم مثلاً نصاً بصورة عمودية:

كود C#

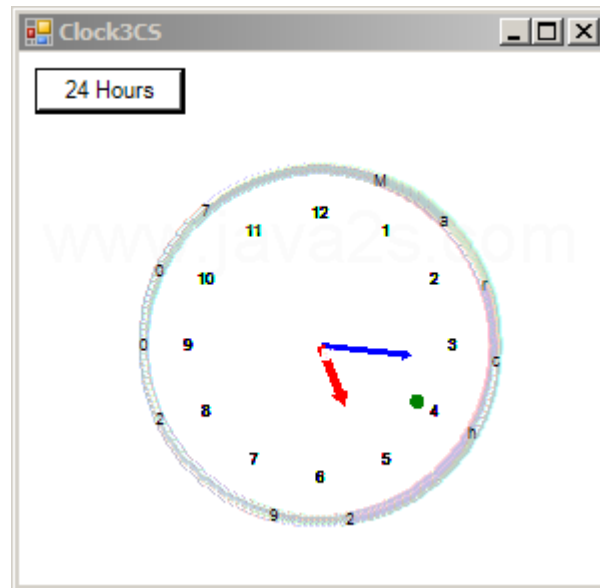
```
Graphics myGraphic = e.Graphics;
StringFormat drawFormat = new StringFormat();
drawFormat.FormatFlags = StringFormatFlags.DirectionVertical;
myGraphic.DrawString("Hello GDI+", new Font("Times New Roman", 30),
Brushes.Red, 100, 20, drawFormat);
```

كود VB.NET

```
Dim myGraphic As Graphics = e.Graphics
Dim drawFormat As New StringFormat()
drawFormat.FormatFlags = StringFormatFlags.DirectionVertical
myGraphic.DrawString("Hello GDI+", New Font("Times New Roman", 30),
Brushes.Red, 100, 20, drawFormat)
```

سيكون الناتج شيئاً مثل هذا:

الصورة 15. 1. رسم النص على النافذة بالكود.

اختم الدرس بمثال جيد ليكون تطبيقاً على دروس الرسوميات لدينا ، المثال موجود على هذا الرابط وهو لرسم ساعة بالشكل التالي:

رابط المثال:

رابط

http://www.java2s.com/Tutorial/CSharp/0480__2D/Clockanimation.htm

هناك الكثير جداً في عالم ال GDI أخشى انني لم استطع إلا وضعك على أول الطريق فيه ، يمكنك الاستزادة منه من هنا .

رابط

http://msdn.microsoft.com/en-us/library/ms533798(VS.85).aspx

الباب

16

Windows Presentation Foundation

1. تعريفات أساسية

1. 1. WPF

Windows Presentation Foundation أو ما يعرف اختصاراً باسم WPF هي تقنية رسومية جديدة من مايكروسوفت بدأت مع net framework 3.0. تعتمد على ما يعرف باسم Extensible Application Markup Language واختصاراً باسم XAML و تنطق (زامل)، وهي تعتبر الجيل الجديد من تقنيات مايكروسوفت في الواجهات المرئية بعد Windows Forms و +GDI و DirectX APIs و Windows Media Player API's للواجهات والرسومات ثنائية وثلاثية الأبعاد والفيديو على الترتيب.

أما مع WPF فقد اصبح بامكانك التحكم بجميع هذه النقاط ابتداء من تصميم الواجهات وانتهاء بالتصميم الثلاثي الأبعاد والفيديو من خلالها.

1. 2. XAML

تهدف XAML إلى الفصل بين الكود والتصميم بصورة كاملة مثلما كان يحدث في صفحات ASP.net ، حيث تعد XAML شبيهة ب HTML الذي تستخدمه لتصميم صفحاتك - مع الفارق - حيث أن ال XAML أقرب إلى نظام لغات البرمجة حيث يحتوي على فئات وخصائص ... الخ.

1. 3. Microsoft Expression

سابقاً ، كان التصميم عبارة عن مجموعة من عمليات الرسوميات ، يتم تعريفها بطريقة ما في الفيجوال ستوديو ، في حين تظل مجرد صور في برنامج لتحرير الصور مثل photoshop وبطريقة أخرى في برنامج مثل Adobe Flash .. تخيل الآن ان لديك برنامج متخصص في الرسوميات ينتج الرسومات بصورة يمكن فهمها في بيئة التطوير الخاصة بك ؟

كانت هذه هي فكرة Expression ، حيث تعتبر برامج متخصصة في التصميم الثابت والمتحرك وخلافه ، ولكن الناتج يمكن أن يكون على شكل XAML يمكن استخدامه في تطبيقاتك المختلفة بسهولة.

هناك عدة اصدارات من هذه البرامج منها المخصص للويب ومنها الرسومي الشبيه بالفوتوشوب او الشبيه بالفلاش . لكننا نتوسع في شرح هذا البرنامج ، ولكن ضع هذا البرنامج في اعتبارك طيلة فترة عملنا على ال WPF حيث يمكن انتاج XAML الذي نستخدمه من خلاله .

كانت هذه مقدمة سريعة جداً ، سنعود لنبدء عالم ال WPF من البداية.

2. أنواع تطبيقات WPF

تطبيق WPF ليس مجرد تطبيق تقليدي فقط ، بل يمكن ان يظهر على أكثر من شكل ، سنحاول التعرف على هذه الأشكال الآن.

1- Traditional Desktop Applications

النوع الأكثر شيوعاً ، تطبيق تقليدي exe ، ال WPF مجرد طريقة لتحسين المظهر والواجهات وخلافه.

2- Navigation-Based

يمكنك من خلال WPF اختيار تطبيقك ليكون تطبيق عادي ولكن لديه خصائص المتصفح لديك Forward و Backward حيث يمكن للمستخدم التنقل بين صفحات تحددها أنت في pages، لا جديد في كونها تطبيق عادي سوى مبدئ التصفح.

3- XBAP Applications

ميزة جديدة من مميزات WPF لبناء تطبيق يعمل من خلال المتصفح ، شبيه جداً بمبدأ ال JAVA Applet، حيث يتم الوصول إليه من خلال عنوان URL ، يقوم بتحميل XBAP application في ال local machine ويقوم بتشغيلها.

Silverlight Application -4

من خلال WPF ايضاً يمكنك بناء تطبيقات يتم استضافتها من خلال المتصفح مباشرة، شديدة الشبه ب embeded flash الموجود في صفحات الإنترنت المختلفة ، يتمتع ال Silverlight بامكانيات غنية في المظهر والتنفيذ على حد سواء.

3. محتويات WPF

System.Windows : العناصر الأساسية لل WPF .

System.Windows.Controls: تحتوي على مجموعة من العناصر اللازمة لبناء تطبيقك مثل القوائم وال Tool Tips وخلافه.

System.Windows.Markup: الفئات الخاصة بفهم وتنفيذ صيغ XAML.

System.Windows.Media: الفئات الخاصة بالتعامل مع الفيديو والصور المتحركة والتصاميم 3D.

System.Windows.Navigation: الفئات الخاصة بالتعامل مع النوع-Navigation Based من التطبيقات الذي وضحناه سابقاً.

System.Windows.Shapes: مجموعة من الأدوات لعمليات التصاميم 2D.

4. تطبيقك الأول في عالم WPF

يمكننا بناء تطبيقات WPF بدون الاعتماد على XAML، ولكننا سنتجاوز هذه المرحلة لنتعلم مباشرة كيفية بناء تطبيقات WPF باستخدام XAML.

لو كنت قد جربت سابقاً التعامل مع HTML فأنت ستكون قد اجتزت نصف المرحلة ، ولمن هذه هي المرة الأولى بالنسبة لهم فقد كانت ال HTML تعتمد على Tags بالشكل التالي مثلاً:

كود	HTML

```
<a href="Ahmed.html">Ahmed</a>
```

الكود السابق يعني رابط Link يشير إلى الصفحة Ahmed.html في حين ان النص المعروض للمستخدم هو Ahmed.

هناك طريقة أخرى لاكواد ال HTML بالشكل التالي مثلاً:

كود	HTML

```
<img src="ahmed.jpg" alt="ahmed picture" border="0" />
```

هذا الكود يعني صورة لديها ToolTip وليس لها حدود ، لو لاحظت فليس لها Tag نهاية.

لو كنت ترغب بمعرفة المزيد عن HTML يمكنك البدء من هنا باللغة العربية :

رابط

http://www.html4arab.com/

ستكون معلومات سريعة + لن تأخذ اكثر من ساعتين في تعلمه + سيكون مفيداً جداً لك في تطوير المواقع + حتى مع استخدام ادوات مثل DreamWaver أو الفيجوال ستوديو في تطبيقاتك فمن المفضل التعرف عليها - شخصياً استخدم لبناء مواقعي ال Notepad في الغالب.

المهم ، هذا هو كل ما نحتاج إليه للبدء في عالم XAML ، بنفس الطريقة سيتم توصيف الأدوات الخاصة بنا ، فهذا زر أمر مثلاً:

كود	XAML

```
<Button Height="80" Width="100">
```

وهذا النوع الآخر:

كود	XAML

```
<Button Height="80" Width="100" Content="ClickMe"/>
```

يمكن ان يكون لدينا زر أمر يحتوي بداخلة على عناصر أخرى ، هذا مثال:

كود	XAML

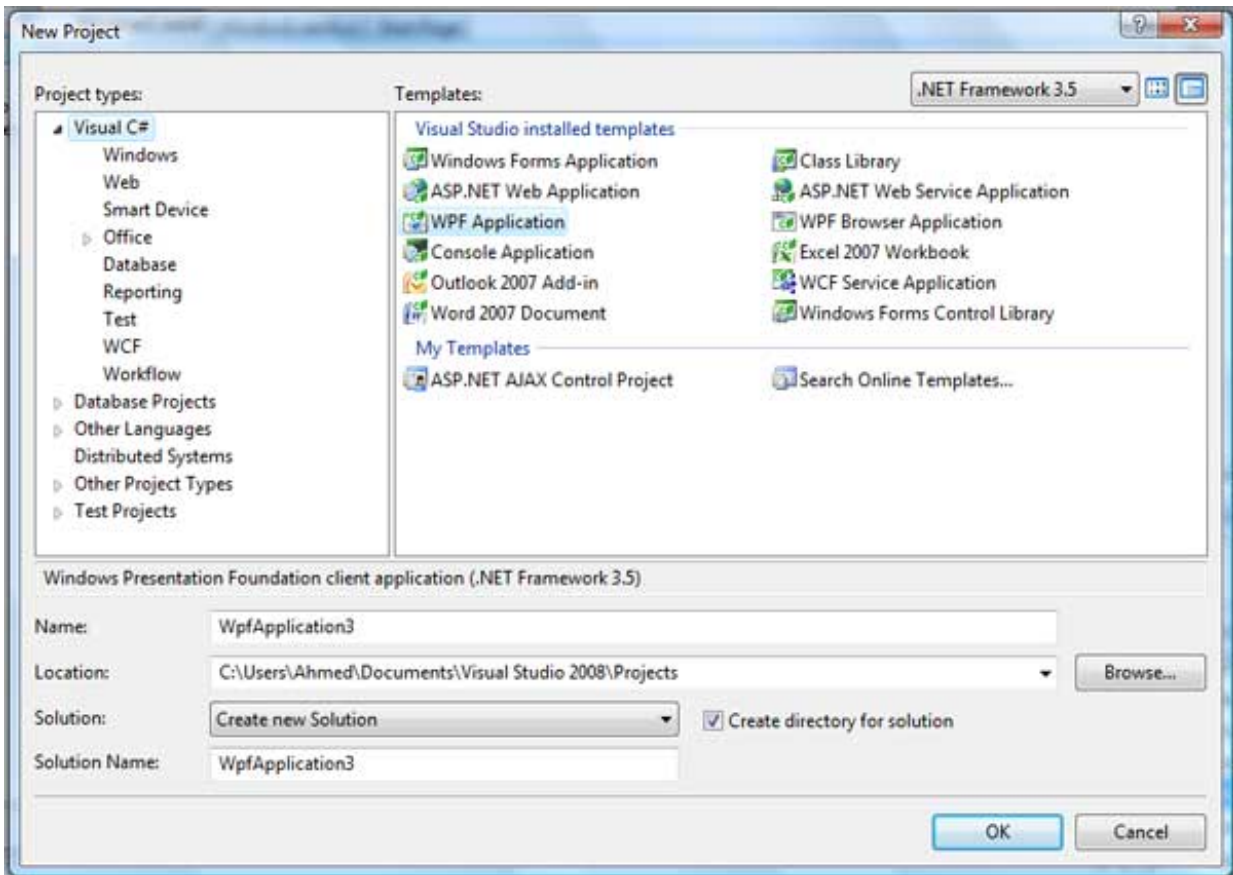
```
<Button Height = "80" Width = "100">
    <Button.Content>
        <ScrollBar Width = "75" Height = "40"/>
    </Button.Content>
</Button>
```

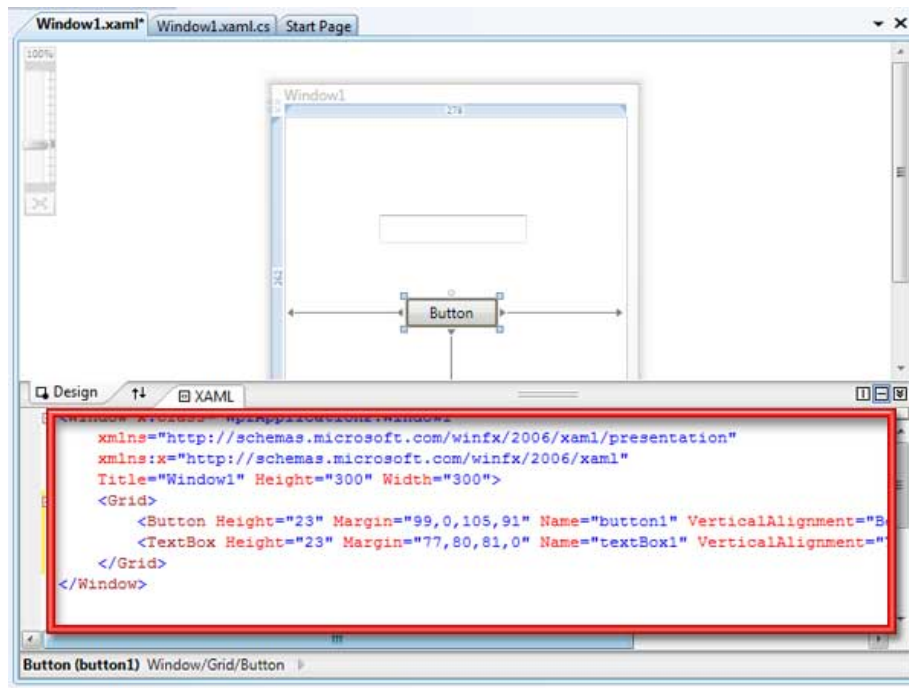
لكن المثال التالي خاطئ نظراً لإن ال Scroll ليست مشتقة من الفئة ContentControl!

كود XAML

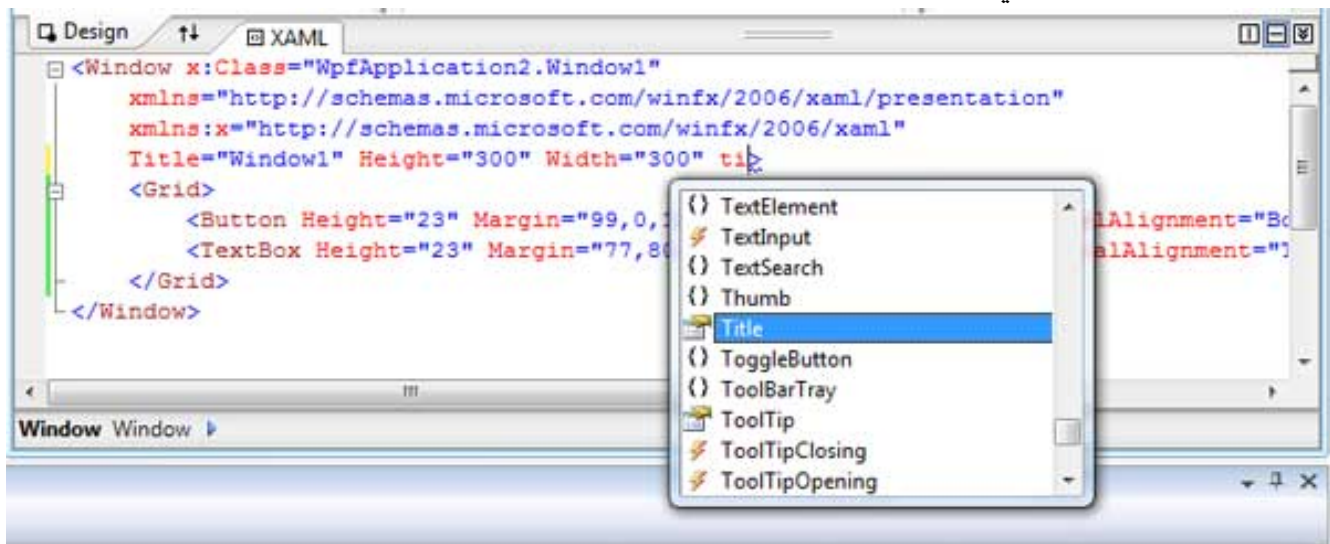
```
<ScrollBar Height = "80" Width = "100">
    <Button Width = "75" Height = "40"/>
</ScrollBar >
```

5. البدء من خلال فيجوال ستوديو

الآن افتح Visual Studio ، قم باختيار انشاء New Project ، قم باختيار WPF Application بالشكل التالي:

الصورة 16. 1. انشاء مشروع WPF جديد.

قم بوضع زر أمر ومربع نص ، ثم لاحظ شاشة ال XAML ، بالتحديد قم بالتركيز على الجزء التالي:

الصورة 16. 2. محرر شفرات ال XAML.

ستجد انه كما ذكرنا بالفعل، قام بعمل انشاء كود XAML لكافة عناصر الفورم ووضعها داخل العنصر Window ثم Grid، ايضاً تجد بعض العناصر الاضافية في حالة التصميم ، الآن اصبح بامكانك كتابة كود XAML او الاعتماد على التصميم كما تفعل مع تطبيقات الويب - إن كنت مبرمج ويب - .

سنحاول الآن التعديل في خصائص Window لتغيير عنوان الفورم ، قم بمسح الخاصية Title، ثم اذهب إلى نهاية الوسم وابدأ بكتابة مسافة ومن ثم نقوم باختيار Title ومن ثم نكتب "WPF= Example" بالشكل التالي مثلاً:

الصورة 16. 3. محرر شفرات ال XAML و تعديل الخصائص من الكود مباشرة.

سيكون الكود الاجمالي XAML بالشكل التالي:

كود XAML

```
<Window x:Class="WpfApplication2.Window1"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Height="300" Width="300" Title="WPF Example">
    <Grid>
        <Button Height="23" Margin="99,0,105,91" Name="button1"
VerticalAlignment="Bottom" Click="button1_Click">Button</Button>
        <TextBox Height="23" Margin="77,80,81,0" Name="textBox1"
VerticalAlignment="Top" />
    </Grid>
</Window>
```

لاحظ ان بامكانك التحكم في جميع الخصائص لكل الأدوات بنفس الطريقة.

الآن جرب كتابة أمر بالضغط على زر الأمر مرتين، ستجد نافذة مستقلة للكود غير تلك التي للتصميم، اكتب فيها الأمر التالي مثلاً:

كود C#

```
private void button1_Click(object sender, RoutedEventArgs e)
{
    textBox1.Text = "Welcome WPF";
}
```

كود VB.NET

```
Private Sub button1_Click(ByVal sender As Object, ByVal e As RoutedEventArgs)
    textBox1.Text = "Welcome WPF"
End Sub
```

والناتج:

الصورة 16. 4. نتيجة التنفيذ.

ايضاً يمكنك كتابة الكود في نفس شاشة التصميم بالشكل التالي:

كود	XAML

```
<x:Code>
    <![CDATA[
void CodeExample(object sender, RoutedEventArgs e)
{
    MessageBox.Show("ahmed");
}

]]>
</x:Code>
```

مع جعل اسم الدالة على الاسم الذي اخترته في الحدث Click لزر الأمر بالشكل التالي مثلاً:

كود	XAML

```
<Button Height="23" Margin="99,0,105,91" Name="button1"
VerticalAlignment="Bottom" Click="CodeExample">Button</Button>
```

ونفس النظام بالنسبة ل VB.net ايضاً أثناء كتابتك للحدث Click يمكنك من خلال Visual Studioانشاء حدث جديد مباشرة...

خاتمة الجزء الأول - نقاط سريعة

- لو كنت تبحث عن معرفة ال Syntax الخاص ب XAML بتوسع يمكنك البدء من هنا:

رابط

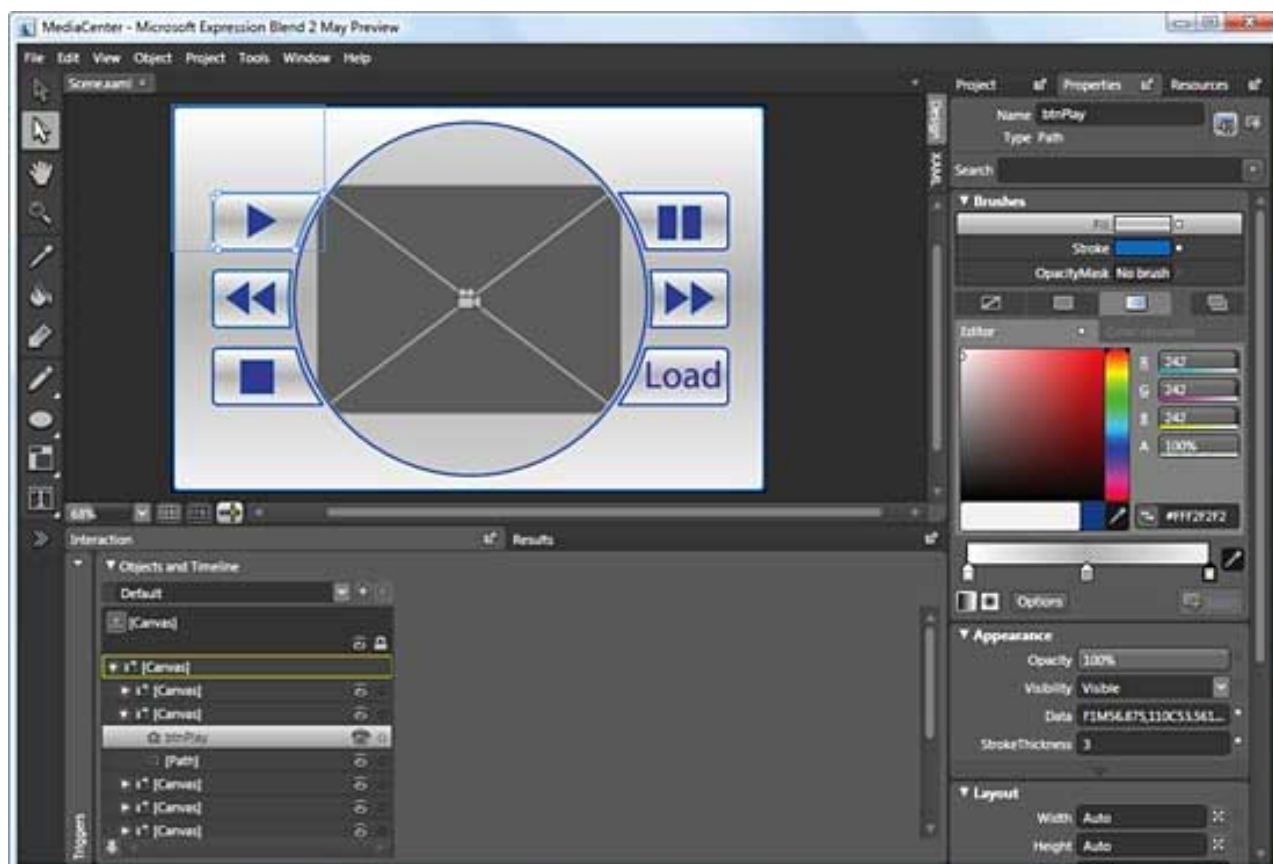
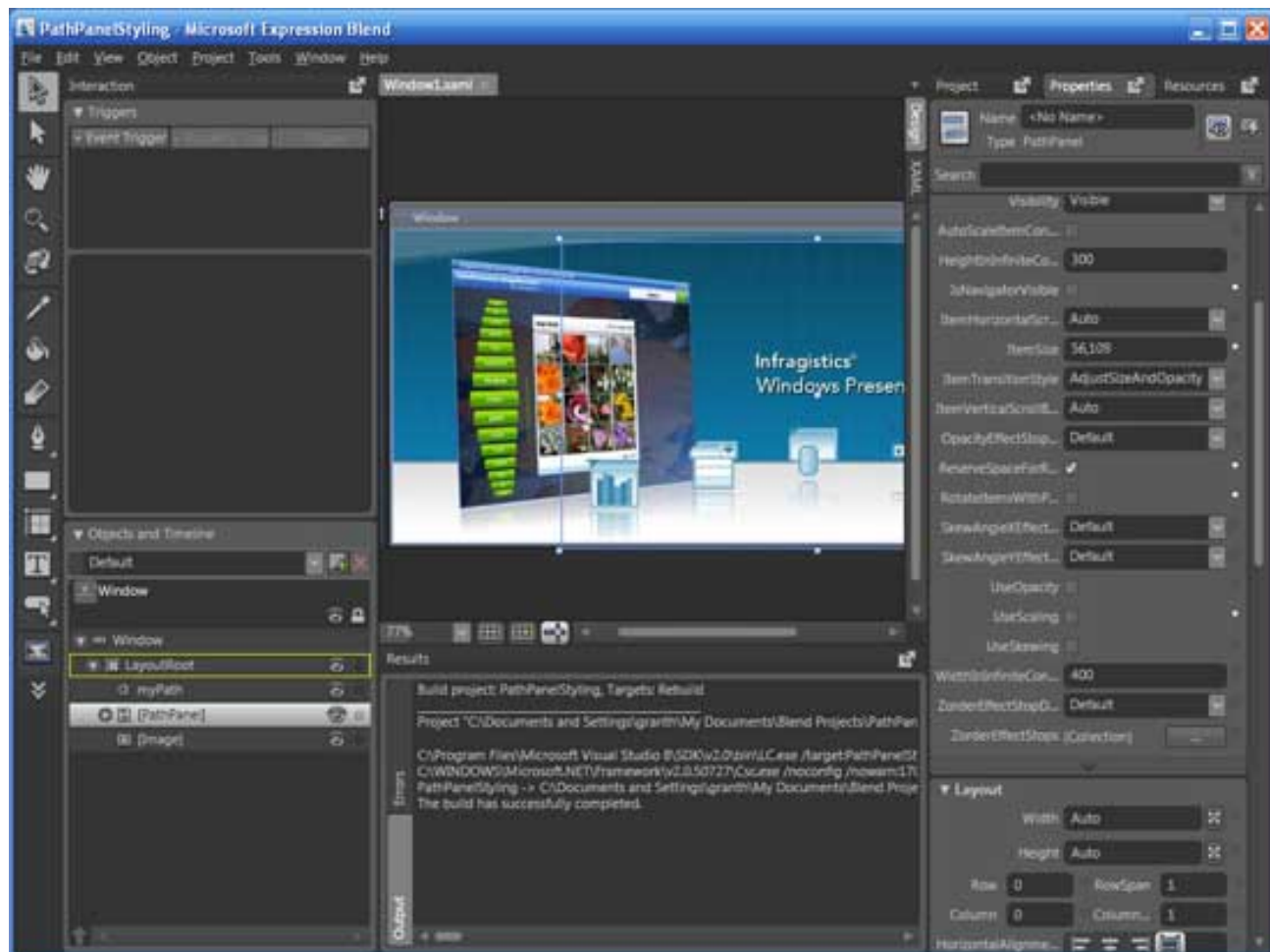
http://msdn.microsoft.com/en-us/library/ms788723.aspx

- لاعتماد أحداث ال FormLoad وخلافه يمكنك اضافتها ل Window بالشكل التالي مثلاً:

كود	XAML

```
Loaded = "Window_Loaded" Closed = "Window_Closed"
```

ونفس الأمر مع جميع أحداث وخصائص كل الأدوات.

- لو استمررت بهذه الطريقة مع WPF فأنت تعطي لنفسك المزيد من التحكم بالمظهر ، لكن لو كنت تريد جمال الواجهات وخلافه فيمكنك البدء مع Microsoft Expression Blend من ضمن مجموعة Microsoft Expression ، هذه صورة للواجهة الرئيسية مثلاً :

الصورة 16. 5. استديو ال Expression Blend.

ناتج تصميمك سيكون XAML يمكنك وضعه في ال Visual Studio مباشرة وسنتعرف عليها في دروس قادمة ...

6. أدوات WPF

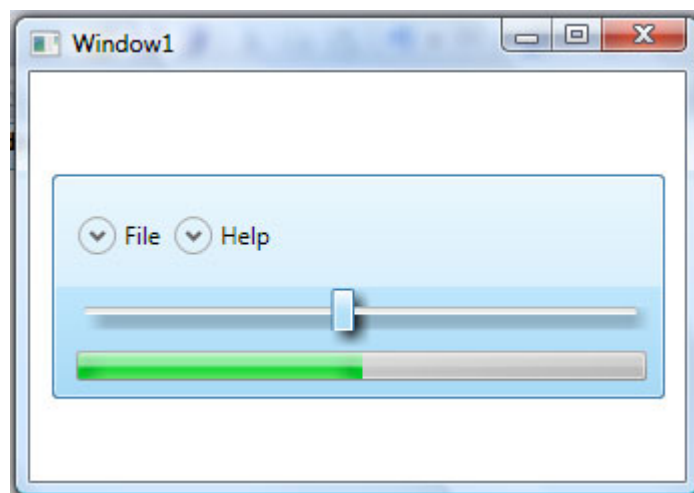
اضافة للادوات التقليدية ، توفر لك WPF مجموعة من الأدوات الجديدة ، أو تغير من مظهرها وخياراتها من أجل استخدامك لها.

أولاً : مجموعة الأدوات التقليدية مجموعة الادوات المعتادة التي تضم Lable, Button وباقي الأدوات العادية ، فقط تتمتع ببعض الخصائص الإضافية لتجميل وتحسين المظهر .

ثانياً : أدوات الفورم مثل القوائم واشرطة التمرير وخلافه.

ثالثاً : أدوات ال media مثل أدوات الصوت والصورة والفيديو وخلافه.

رابعاً : أدوات المظهر تضم بعض الادوات الاساسية مثل ال Groupbox وال Panel وأخرى جديدة مثل Canvas وStackPanel .

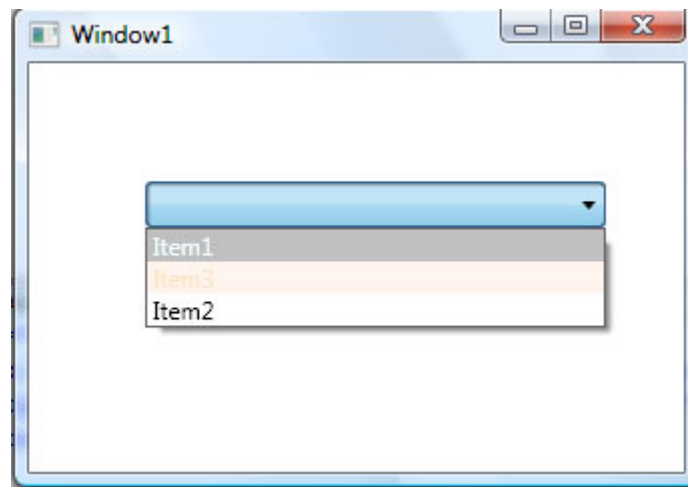
الصورة توضح مجموعة من أدوات WPF المختلفة

الصورة 16 .6. تركيب عدة أدوات WPF مع بعضها البعض.

وكود ال XAML الخاص بها بالشكل التالي:

كود	XAML

```
<Window x:Class="WpfApplication3.Window1"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Window1" Height="240" Width="343">
    <Grid>
        <Button Name="btnPurchaseOptions" Margin="10.5,51,10.5,41">
            <StackPanel>
                <StackPanel Orientation = "Horizontal"></StackPanel>
            </StackPanel>
        </Button>
        <Expander Name="helpExpander" Header="Help" Height="23"
HorizontalAlignment="Left" Margin="70,70,0,0" VerticalAlignment="Top"
Width="51.757"></Expander>
        <Expander Name="fileExpander" Header="File" Height="23"
HorizontalAlignment="Left" Margin="22,70,0,0" VerticalAlignment="Top"
Width="44.95"></Expander>
        <Slider Height="21" Margin="21.757,0,19,75" Name="slider1"
VerticalAlignment="Bottom">
            <Slider.BitmapEffect>
                <DropShadowBitmapEffect />
            </Slider.BitmapEffect>
        </Slider>
        <ProgressBar Height="15" Margin="23,0,19,50" Name="progressBar1"
VerticalAlignment="Bottom" Value="50" />
    </Grid>
</Window>
```

نقاط سريعة حول أدوات WPF

- الأشكال البيضاوية: Ellipse لتحديد الأشكال البيضاوية والدوائر وخلافه.
- عناصر List أو ال ComboBox اصبح بالامكان تحديد لون وخلفية مختلفة لكل منها بالشكل التالي مثلاً:

الصورة 16. 7. التحكم في عناصر ال ComboBox.

- يمكن تفعيل خاصية تصحيح الخطأ SpellCheck في أي مربع نص بالشكل التالي مثلاً:

كود XAML

```
<TextBox SpellCheck.IsEnabled="True" Height="23" Margin="20,44,16,0"
Name="textBox1" VerticalAlignment="Top" Grid.Row="10" />
```

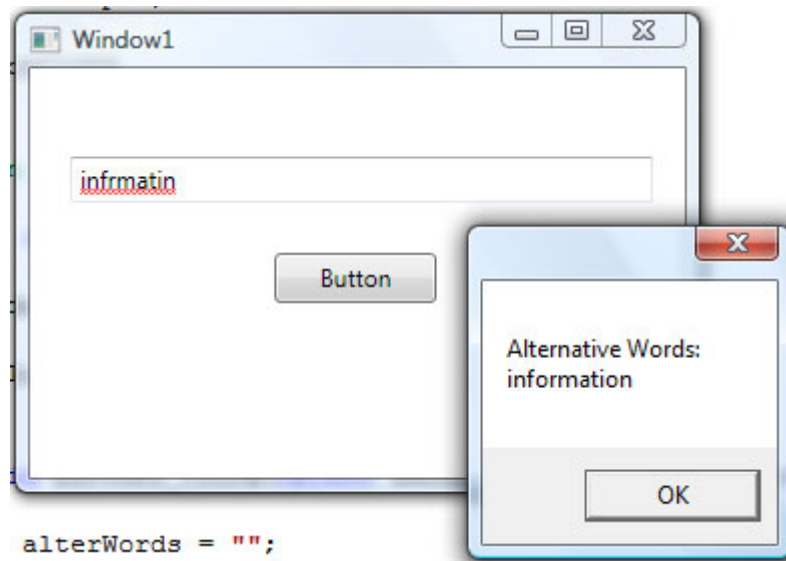
ويمكن معرفة البدائل للخطأ الموجود عن طريق كتابة كود مثل التالي:

كود C#

```
string alterWords = "";
SpellingError error = textBox1.GetSpellingError(0);
if (error != null)
{
    foreach (string s in error.Suggestions)
    {
        alterWords += s + "\n";
    }
    MessageBox.Show("Alternative Words:\n" + alterWords);
}
```

كود VB.NET

```
Dim alterWords As String = ""
Dim [error] As SpellingError = textBox1.GetSpellingError(0)
If [error] IsNot Nothing Then

  For Each s As String In [error].Suggestions
      alterWords += s + "" & Chr(10) & ""
  Next

  MessageBox.Show("Alternative Words:" & Chr(10) & "" + alterWords)
End If
```

والناتج سيكون شيئاً مثل هذا

الصورة 16. 8. استغلال الخاصية Spell Cheker مع ال TextBox.

7. ربط البيانات Data-Binding

في ال WPF ، نستطيع ربط قيم عناصر بطريقة مباشرة ، مثلاً لربط قيم Slider بـ TextBox مباشرة ، نقوم باضافة فقط الجزء التالي للأداة المراد ربطها:

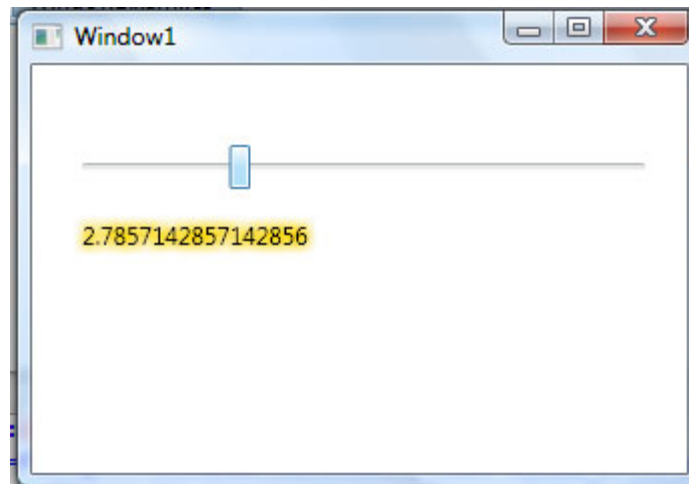
كود	XAML

```
DataContext = "{Binding ElementName=slider1}" Content = "{Binding Path=Value}"
```

بالتالي يصبح كود XAML الكامل بالشكل التالي:

كود	XAML

```
<Slider Height="27" Margin="20,40,16,0" Name="slider1" VerticalAlignment="Top"
/>
<Label Height="28" DataContext = "{Binding ElementName=slider1}" Content =
"{Binding Path=Value}" Margin="20,72,16,0" Name="label1"
VerticalAlignment="Top" BorderThickness="0">
```

والناتج:

الصورة 16. 9. استغلال خاصية ربط البيانات.

8. WPF 2D

دنعود مرة أخرى لعالم الرسوميات ثنائية البعد ولكن في ال WPF هذه المرة ، سنتعرف على تقنيات شبيهة بتلك التي تعاملنا معها في GDI+ ولكن هذه المرة من خلال WPF.

بداية تحتوي خيارات الرسم على واحد من الفئات التالية:

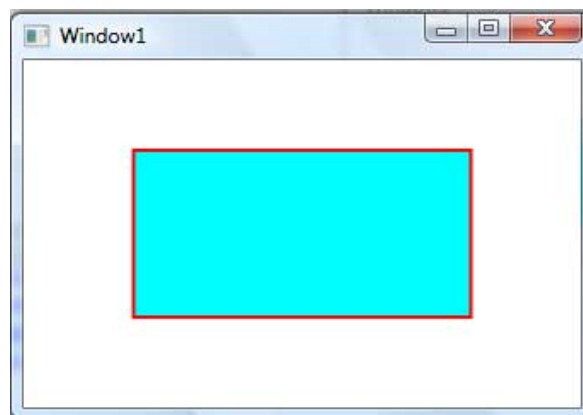
```
System.Windows.Shapes
System.Windows.Media.Drawing
System.Windows.Media.Visual
```

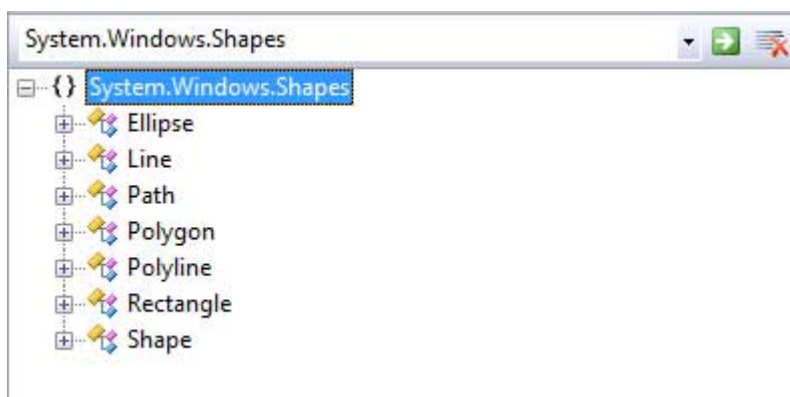
8. 1. الرسم باستخدام Shapes

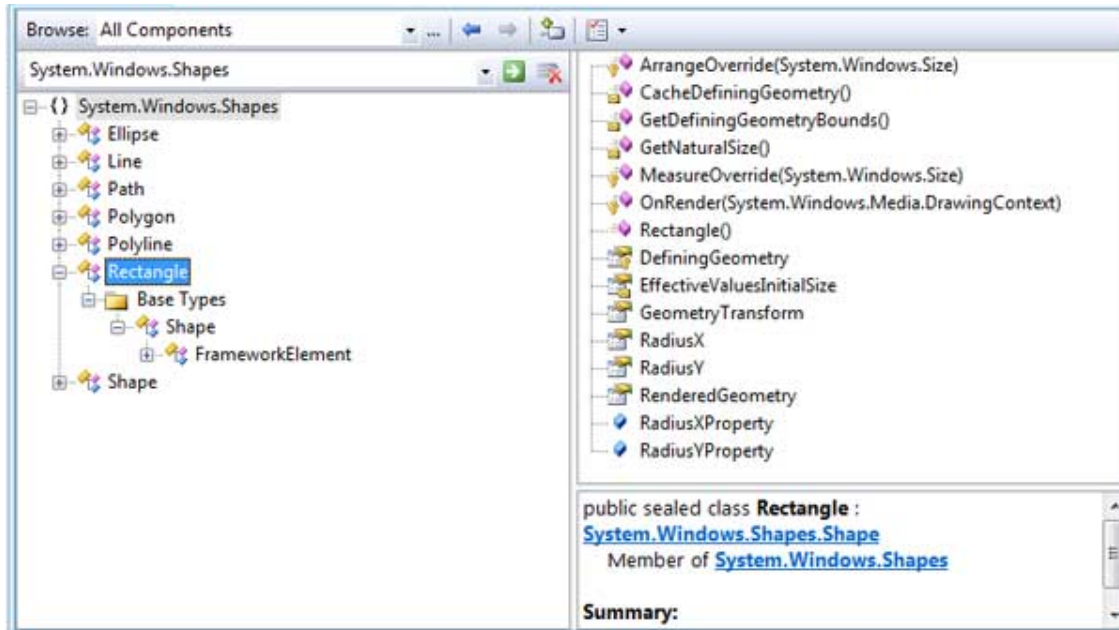
تقع جميع الاشكال تحت الفئة System.Windows.Shapes ، نبدأ بابسط مثال لرسم مستطيل مثلاً

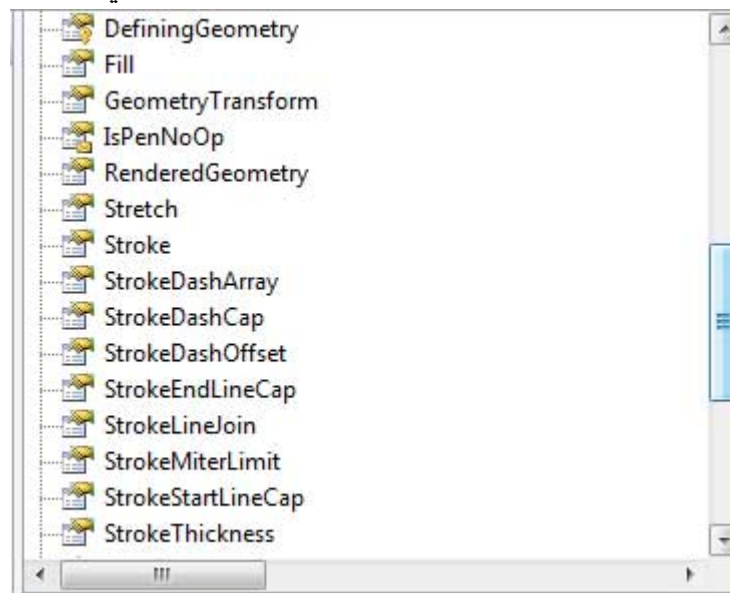
كود	XAML

```xml
<Rectangle Height="100" Width="200" Stroke="Red" StrokeThickness="2"
Fill="Aqua" />
```

الناتج سيكون بالشكل التالي:

الصورة 16. 10. رسم مستطيل مع خاصية الحواف و التلوين.

الأشكال التي يمكنك رسمها يمكنك الوصول إليها عن طريق Object Browser بالشكل التالي مثلاً

الصورة 16. 11. متصفح ال Object Browser.

ويمكننا التعرف على الخصائص التي يمكن ان نجدها لأي شكل منهم ، الشكل التالي كمثال:

الصورة 16. 12. متصفح ال Object Browser.

كما يمكنك التعرف على الخصائص العامة لل Shape الموجودة في الصورة التالية مثلاً:

الصورة 16. 13. خصائص ال Shape.

8 .2. خصائص القلم Pen

يتم تعريف القلم المستخدم في اي عملية رسم بالصورة التالية مثلاً:

كود XAML
```
<Pen Thickness="10" LineJoin="Round" EndLineCap="Triangle" StartLineCap="Round"
/>
```

شبيه جداً لو لاحظت بما تعلمناه في +GDI.

8 .3. خصائص الفرشاة Brush

هناك عدة انوع من الفرش يمكن استخدامها في تطبيقاتك ، منها:

DrawingBrush: تلوين عادي.

ImageBrush: تلوين جزء من الشكل بجزء من صورة.

LinearGradientBrush: تدرج خطي لعدة الون.

RadialGradientBrush: تدرج دائري لعدة الوان.

SolidColorBrush: فرشاة للون واحد فقط.

أبسط مثال عليها هي الفرشاة Solid بلون واحد فقط:

كود XAML
```
<Ellipse Height ="50" Width ="50">
    <Ellipse.Fill>
        <SolidColorBrush Color ="Aqua"/>
    </Ellipse.Fill>
</Ellipse>
```

مثال باستخدام RadialGradientBrush للتدرج الدائري:

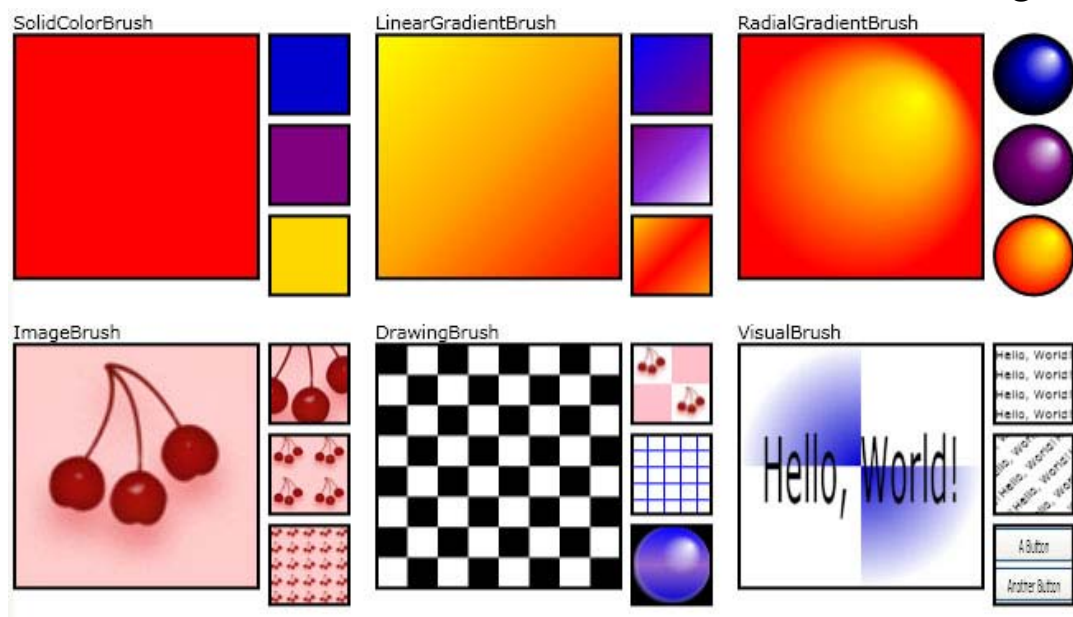
كود	XAML

```
<Ellipse Width ="75" HorizontalAlignment="Left" Margin="28,30,0,96">
    <Ellipse.Fill>
        <RadialGradientBrush GradientOrigin="0.5,0.5" Center="0.5,0.5"
RadiusX="0.5" RadiusY="0.5">

            <GradientStop Color="Yellow" Offset="0" />
            <GradientStop Color="Red" Offset="0.25" />
            <GradientStop Color="Blue" Offset="0.75" />
            <GradientStop Color="LimeGreen" Offset="1" />
        </RadialGradientBrush>
    </Ellipse.Fill>
</Ellipse>
```

وأخيراً مثال لطباعة جزء من صورة داخل شكل:

كود	XAML

```
<Rectangle Height ="100" Width ="300">
    <Rectangle.Fill>
        <ImageBrush>
            <ImageBrush.ImageSource>
                <BitmapImage UriSource ="pic.JPG"/>
            </ImageBrush.ImageSource>
        </ImageBrush>
    </Rectangle.Fill>
</Rectangle>
```

ناتج من موقع مايكروسوفت لاستخدامات الفرش المختلفة:

الصورة 16. 14. نتائج استعمال الفرشاه Brush.

وهذا الرابط لمزيد من التفاصيل :

 رابط

http://msdn.microsoft.com/en-us/library/aa970904.aspx

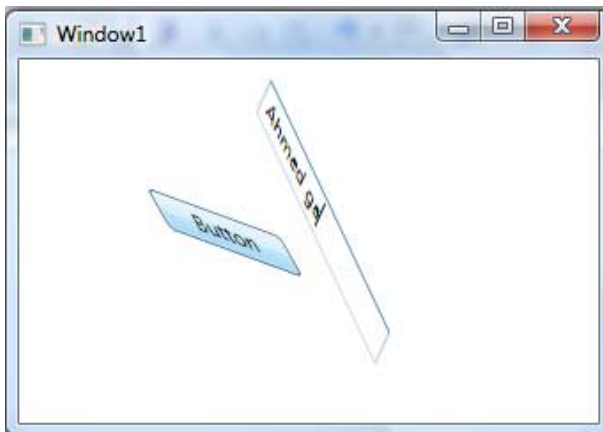
8. 4. ال Transformations

ستفيدك كثيراً هذه الخصائص في حالة برمجة العناصر المتحركة او المتأثرة بمدخلات المستخدم، حيث يمكنك تحريك الأشكال وتدويرها وعكسها وخلافه من العمليات المعروفة.

المثال التالي يوضح بعض التأثيرات على مجموعة من الأدوات الموجودة لدينا في الفورم :

كود XAML

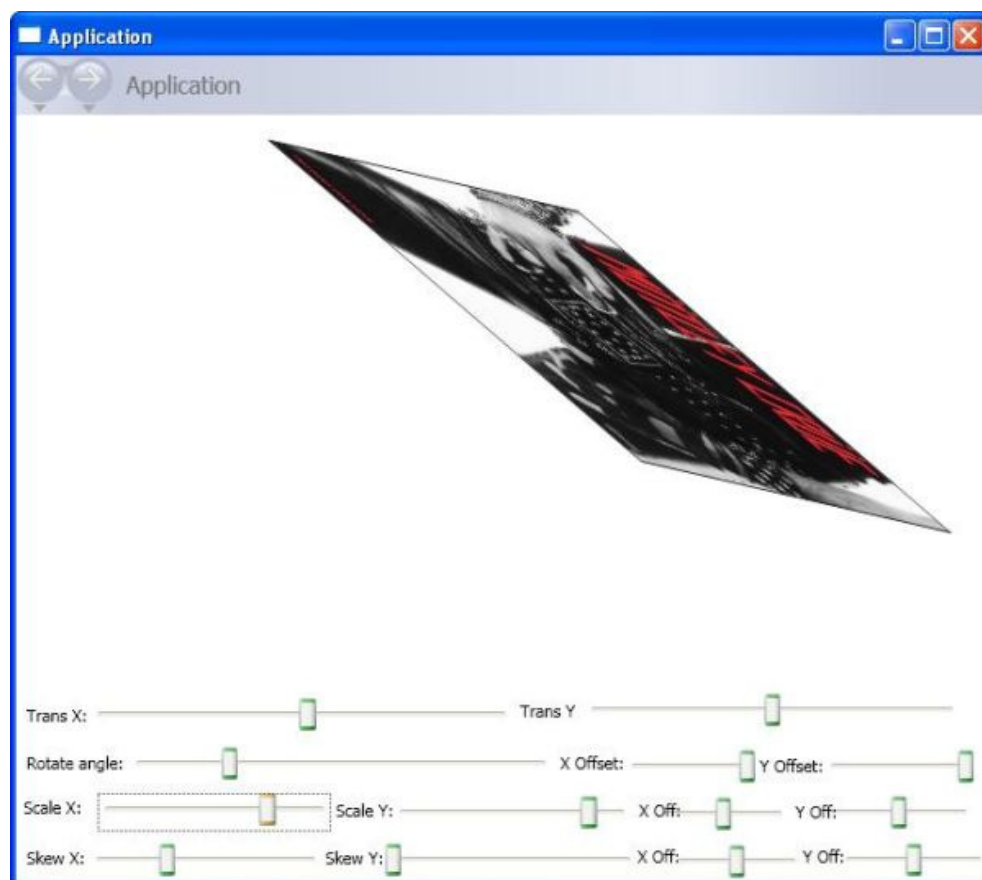
```
<Button Height="23" Margin="72,72,0,0" Name="button1" VerticalAlignment="Top"
HorizontalAlignment="Left" Width="75">Button
    <Button.RenderTransform>
        <SkewTransform AngleX ="30" AngleY ="20"/>
    </Button.RenderTransform>
</Button>
<TextBox Margin="142,11,59,0" Name="textBox1" Height="23"
VerticalAlignment="Top">Ahmed
    <TextBox.RenderTransform>
        <TransformGroup>
            <SkewTransform AngleX ="20" AngleY ="20"/>
            <RotateTransform Angle ="45"/>
            <SkewTransform AngleX ="5" AngleY ="20"/>
        </TransformGroup>
    </TextBox.RenderTransform>
</TextBox>
```

وهذه صورة للناتج:

الصورة 16. 15. نتائج عملية استعمال ال Transformation.

على هذا الرابط تطبيق WPF جميل جداً للتأثير بصور مختلفة على اي شكل:

رابط

http://www.codeproject.com/KB/WPF/TransformationsIntro.aspx

هذه صورة التطبيق:

الصورة 16. 16. برنامج المثال الذي ستقوم بتنزيله من موقع ال MSDN.

9. ال Animation في WPF

تقع كافة الخصائص والدوال المتعلقة بعمليات التحريك وال Animation تحت مجال الأسماء System.Windows.Media.Animation، اي حركة تحصل لأي اداة لديك لا بد لها من ثلاث خصائص From، To، By حيث تحدد نقطة البداية والنهاية والخاصية التي يتم فيها التحريك ، وهو ما سنتعرف عليه لاحقاً...

ولأي حركة هناك ايضاً Timeline، اهم عناصره هي:

العنصر	الوصف
AccelerationRatio, DecelerationRatio	للتحكم في سرعة الحركة
AutoReverse	للعودة للخلف بعد انتهاء الحركة
BeginTime This	الوقت الذي تبدأ بعده الحركة، القيمة 0 تعني البدء المباشر
Duration	الفترة التي تستغرقها عملية الحركة
FillBehavior, RepeatBehavior	تحديد ماذا سيحدث بعد انتهاء الحركة سواء الاعادة أو خلافه

الجدول 16 .1. أهم خصائص الفئة Timeline

مثال، تغيير حجم الخط في Label :

كود C#

```
DoubleAnimation dblAnim = new DoubleAnimation();
dblAnim.From = 10;
dblAnim.To = 30;
label1.BeginAnimation(Label.FontSizeProperty, dblAnim);
```

كود VB.NET

```
Dim dblAnim As New DoubleAnimation()
dblAnim.From = 10
dblAnim.[To] = 30
label1.BeginAnimation(Label.FontSizeProperty, dblAnim)
```

بداية الحركة:

نهاية الحركة:

يمكن تحديد مدة الحركة بالشكل التالي مثلاً

كود C#
```
dblAnim.Duration = new Duration(TimeSpan.FromSeconds(4));
```

كود VB.NET
```
dblAnim.Duration = new Duration(TimeSpan.FromSeconds(4))
```

ولعكس الحركة بعد الانتهاء:

كود C#
```
dblAnim.AutoReverse = true;
```

كود VB.NET
```
dblAnim.AutoReverse = true
```

ويمكننا تحديد اعادة العرض بعد انتهاءه:

كود C#
```
dblAnim.RepeatBehavior = RepeatBehavior.Forever;
```

كود VB.NET
```
dblAnim.RepeatBehavior = RepeatBehavior.Forever
```

او اعادته لاربع مرات فقط على سبيل المثال :

كود C#
```
dblAnim.RepeatBehavior = new RepeatBehavior(4);
```

كود VB.NET
```
dblAnim.RepeatBehavior = new RepeatBehavior(4)
```

10. الحركة باستخدام XAML

كما اتفقنا منذ اللحظة الأولى في هذه الدروس، انه بالامكان تطبيق اي من الاوامر عن طريق XAML او عن طريق الكود كون جميع هذه الخصائص يمكن الوصول إليها من الكود والعكس، الآن لدينا مثال حول جعل Button يقوم بالدوران حول نفسه في حالة الضغط عليه بالماوس - من كتاب Pro CSharp 2008 :

كود XAML

```
<Button Name="myAnimatedButton" Width="120" Height = "40"
RenderTransformOrigin="0.5,0.5" Content = "OK">
    <Button.RenderTransform>
        <RotateTransform Angle="0"/>
    </Button.RenderTransform>
    <!-- The animation is triggered when the button is clicked -->
    <Button.Triggers>
        <EventTrigger RoutedEvent="Button.Click">
            <BeginStoryboard>
                <Storyboard>
                    <DoubleAnimationUsingKeyFrames
Storyboard.TargetName="myAnimatedButton"
Storyboard.TargetProperty=
"(Button.RenderTransform).(RotateTransform.Angle)"
Duration="0:0:2" FillBehavior="Stop">
                        <DoubleAnimationUsingKeyFrames.KeyFrames>
                            <LinearDoubleKeyFrame Value="360" KeyTime="0:0:1"
/>
                            <DiscreteDoubleKeyFrame Value="180"
KeyTime="0:0:1.5" />
                        </DoubleAnimationUsingKeyFrames.KeyFrames>
                    </DoubleAnimationUsingKeyFrames>
                </Storyboard>
            </BeginStoryboard>
        </EventTrigger>
    </Button.Triggers>
</Button>
```

11. تعريف Styles

إذا كنت قد جربت سابقاً اي نوع من برمجة الويب كنت ستعرف ان لدينا ما يعرف باسم styles وهي مجموعة من الخصائص تحدد الطول والعرض والالوان وخلافه تحت مسمى مثلاً DarkStyle، بحيث يمكن بعد ذلك استخدامها في اي اداة بكتابة اسم ال Style فقط.

هذا هو ما نحتاج إليه أيضاً في ال WPF حيث اننا لن نقوم بكتابة كل هذا الحجم من التوصيف لكل زر امر مثلاً في حالة أن لدينا عدة ازرار أمر لها نفس ال Style، لذا سنقوم بتعريف Style بالشكل التالي مثلاً:

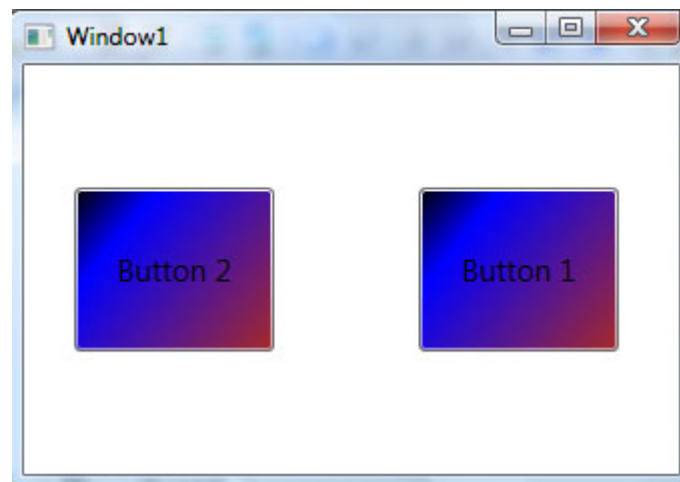
كود	XAML

```
<Window.Resources>
    <Style x:Key ="darkstyle">
        <Setter Property ="Button.FontSize" Value ="15"/>
        <Setter Property ="Button.Background">
            <Setter.Value>
                <LinearGradientBrush StartPoint="0,0" EndPoint="1,1">
                    <GradientStop Color="Black" Offset="0" />
                    <GradientStop Color="Blue" Offset="0.25" />
                    <GradientStop Color="Brown" Offset="1" />
                </LinearGradientBrush>
            </Setter.Value>
        </Setter>
    </Style>
</Window.Resources>
```

والآن لكل زر أمر يكفي التعريف بالشكل التالي:

كود	XAML

```
<Button Name="b1" Width = "100"
Style ="{StaticResource darkstyle}" Content = "Button 1"
HorizontalAlignment="Right" Margin="0,61,30,61" />
<Button Name="b2" Width = "100"
Style ="{StaticResource darkstyle}" Content = "Button 2"
HorizontalAlignment="Left" Margin="25,61,0,61" />
```

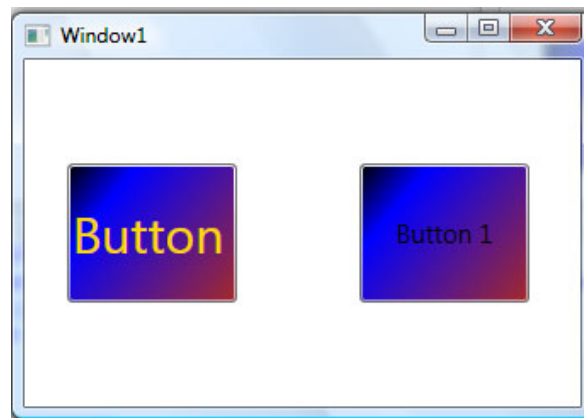
الناتج سيكون بالشكل التالي:

الصورة 16. 17. نتائج التنفيذ.

11 .1. تغيير طبيعة ال Style

يمكنك تعريف اشياء مخصصة لتغيير الموجود في ال Style المستخدم، مثلاً لتحديد زر امر حجم الخط ولونه فيه مختلف لا يلزم تغيير ال Style بالكامل بل يكفي كتابته بالشكل التالي:

كود	XAML

```
<Button Name="b2" Width = "100"
Style ="{StaticResource darkstyle}" Content = "Button 2"
HorizontalAlignment="Left" Margin="25,61,0,61" FontSize="30" />
```

والناتج:

الصورة 16. 18. نتائج التنفيذ.

11 .2. اشتقاق Style من آخر

لعمل Style جديد يتم أخذه من Style قديم يمكن كتابة تعريفه بالشكل التالي:

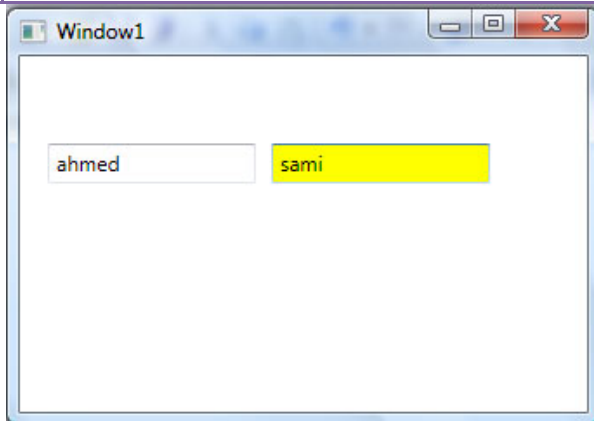
كود	XAML

```
<Style x:Key ="darkredstyle" BasedOn = "{StaticResource darkstyle}">
```

11 .3. تصميم Style باستخدام Triggers

يمكنك ال Triggers من تحديد حالات للاداة مرتبطة بحدث مرور الماوس او غيره ، فمثلاً لتلوين مربع نص بلون مختلف عند حصوله على التحديد نكتب Style بالشكل التالي:

كود	XAML

```
<Window.Resources>
    <Style x:Key ="txtstyle" TargetType = "{x:Type TextBox}">
        <Setter Property = "Background" Value = "White"/>

        <Style.Triggers>
            <Trigger Property = "IsFocused" Value = "True">
                <Setter Property = "Background" Value = "Yellow"/>
            </Trigger>
        </Style.Triggers>
    </Style>
</Window.Resources>
```

والناتج لمربع النص الذي عليه التحديد:

الصورة 16. 19. نتائج التنفيذ.

برمجياً

مثال منقول من Pro CSharp 2008، اضافة عناصر للقائمة بها مجموعة من ال Styles ومن ثم تحديد Style زر الامر ليحتوي على أحدها بالشكل التالي مثلاً:

كود	C#

```
public partial class MainWindow : Window
{
    public MainWindow()
    {
        InitializeComponent();
        // Add items to our list box.
        lstStyles.Items.Add("TiltStyle");
        lstStyles.Items.Add("GreenStyle");
        lstStyles.Items.Add("MouseOverStyle");
    }
    protected void comboStyles_Changed(object sender, RoutedEventArgs args)
    {
        // Get the selected style name from the list box.
        System.Windows.Style currStyle = (System.Windows.Style)
        FindResource(lstStyles.SelectedValue);
        // Set the style of the button type.
        this.btnMouseOverStyle.Style = currStyle;
    }
}
```

VB.NET	كود

```
Partial Public Class MainWindow
    Inherits Window
    Public Sub New()
        InitializeComponent()
        ' Add items to our list box.
        lstStyles.Items.Add("TiltStyle")
        lstStyles.Items.Add("GreenStyle")
        lstStyles.Items.Add("MouseOverStyle")
    End Sub
    Protected Sub comboStyles_Changed(ByVal sender As Object, ByVal args As
RoutedEventArgs)
        ' Get the selected style name from the list box.
        Dim currStyle As System.Windows.Style =
DirectCast(FindResource(lstStyles.SelectedValue), System.Windows.Style)
        ' Set the style of the button type.
        Me.btnMouseOverStyle.Style = currStyle
    End Sub
End Class
```

والناتج:

C#	كود

```
public partial class MainWindow : Window
{
    public MainWindow()
    {
        InitializeComponent();
        // Add items to our list box.
        lstStyles.Items.Add("TiltStyle");
        lstStyles.Items.Add("GreenStyle");
        lstStyles.Items.Add("MouseOverStyle");
    }
    protected void comboStyles_Changed(object sender, RoutedEventArgs args)
    {
        // Get the selected style name from the list box.
        System.Windows.Style currStyle = (System.Windows.Style)
        FindResource(lstStyles.SelectedValue);
        // Set the style of the button type.
        this.btnMouseOverStyle.Style = currStyle;
    }
}
```

VB.NET	كود

```
Partial Public Class MainWindow
    Inherits Window
    Public Sub New()
        InitializeComponent()
        ' Add items to our list box.
        lstStyles.Items.Add("TiltStyle")
        lstStyles.Items.Add("GreenStyle")
        lstStyles.Items.Add("MouseOverStyle")
    End Sub
    Protected Sub comboStyles_Changed(ByVal sender As Object, ByVal args As
RoutedEventArgs)
        ' Get the selected style name from the list box.
        Dim currStyle As System.Windows.Style =
DirectCast(FindResource(lstStyles.SelectedValue), System.Windows.Style)
        ' Set the style of the button type.
        Me.btnMouseOverStyle.Style = currStyle
    End Sub
End Class
```

الصورة 16. 20. نتائج التنفيذ.

12. الـ Templates

تستخدم ايضاً لتغيير خصائص الأدوات، ولكن الفارق بينها وبين Styles انها تستطيع ان تلغي بالكامل الطبيعة الاساسية للأداة ، مثلاً هذه ال Templates:

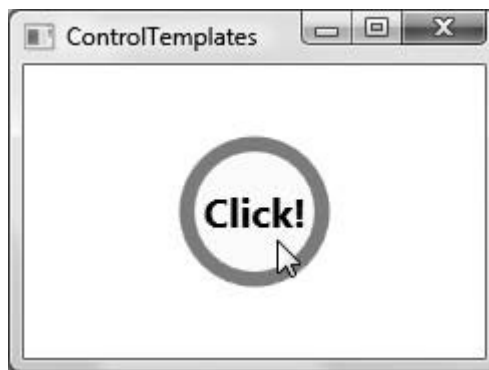
XAML	كود

```
<Grid.Resources>
    <!-- A simple template for a round button for items in this grid -->
    <ControlTemplate x:Key ="roundButtonTemplate" TargetType ="{x:Type
Button}">
        <Grid>
            <Ellipse Name ="OuterRing" Width ="75" Height ="75" Fill
="DarkGreen"/>
            <Ellipse Name ="InnerRing" Width ="60" Height ="60" Fill
="MintCream"/>
            <ContentPresenter HorizontalAlignment="Center"
VerticalAlignment="Center"/>
        </Grid>
    </ControlTemplate>
</Grid.Resources>
```

والآن يمكنك تحديدها لزر أمر مثلاً:

XAML	كود

```
<Button Name ="myButton" Foreground ="Black" FontSize ="20" FontWeight ="Bold"
Template ="{StaticResource roundButtonTemplate}"
Click ="myButton_Click">Click!
</Button>
```

والناتج:

الصورة 16. 21. نتائج التنفيذ.

13. 3D WPF

بعد ان تعلمنا كل ما نحتاج إليه للبدء بالغوص في عالم WPF من وجهة نظر ثنائية الابعاد، سنحاول في هذا الدرس التعرف على اساسيات العالم الثلاثي الأبعاد في WPF او ما يسمى بـ Avalon .

يختلف عالم ال 3D كثيراً عن عالم ال 2D، حيث تحتاج لبعض مفاهيم الهندسة الفراغية أولاً ومن ثم معرفة بعض المفاهيم مثل Projection وخلافه ، وبما اننا لن نتخصص هنا في مجال ال Graphics يمكنك البدء بمعرفة ما تريد من خلال هذه المقدمة السريعة :

رابط

http://developer.nvidia.com/docs/IO/11278/Intro-to-Graphics.pdf

الآن سنبدأ مع WPF ...

لو كانت لديك بعض الخبرة مع علم ال Graphics فأنت تدرك ان الهرم - الشكل الاساسي دوماً في عالم 3D والتطبيق الذي يشابه Hello World في عالم ال Graphics - ما هو إلا مجموعة من الخطوط بين نقاط مختلفة، او تستطيع القول انها مجموعة من المثلثات تمثل عدد أوجه الهرم - 3 اوجه - ولكنك تحتاج لفهم نظريات ال Projection لتحديد الابعاد والقياسات المناسبة لرسم هذه المثلثات في حالة ال 3D بحيث تظهر بصورة طبيعية.

بداية سنقوم باستيراد الفئة System.Windows.Media.Media3D وهي المختصة بالتعامل مع هذا العالم.

C#	كود

```
using System.Windows.Media.Media3D;
```

VB.NET	كود

```
Imports System.Windows.Media.Media3D;
```

سنحتاج اولاً لتعريف Viewport3D والذي يمكننا من تحديد مجال الرؤية ونوع الكاميرا وخلافه، بالشكل التالي مثلاً:

كود XAML

```
    <Viewport3D Name="mainViewport" ClipToBounds="True">
        <Viewport3D.Camera>
            <PerspectiveCamera
FarPlaneDistance="100"
LookDirection="-11,-10,-9"
UpDirection="0,1,0"
NearPlaneDistance="1"
Position="11,10,9"
FieldOfView="70" />
        </Viewport3D.Camera>
        <ModelVisual3D>
            <ModelVisual3D.Content>
                <DirectionalLight
 Color="White"
 Direction="-2,-3,-1" />
            </ModelVisual3D.Content>
        </ModelVisual3D>
    </Viewport3D>
```

وفي زر الأمر:

كود C#

```
MeshGeometry3D triangleMesh = new MeshGeometry3D();
Point3D point0 = new Point3D(0, 0, 0);
Point3D point1 = new Point3D(5, 0, 0);
Point3D point2 = new Point3D(0, 0, 5);
triangleMesh.Positions.Add(point0);
triangleMesh.Positions.Add(point1);
triangleMesh.Positions.Add(point2);
triangleMesh.TriangleIndices.Add(0);
triangleMesh.TriangleIndices.Add(2);
triangleMesh.TriangleIndices.Add(1);
Vector3D normal = new Vector3D(0, 1, 0);
triangleMesh.Normals.Add(normal);
triangleMesh.Normals.Add(normal);
triangleMesh.Normals.Add(normal);
Material material = new DiffuseMaterial(
    new SolidColorBrush(Colors.DarkKhaki));
GeometryModel3D triangleModel = new GeometryModel3D(
    triangleMesh, material);
ModelVisual3D model = new ModelVisual3D();
model.Content = triangleModel;
this.mainViewport.Children.Add(model);
```

كود	VB.NET

```
Dim triangleMesh As New MeshGeometry3D()
Dim point0 As New Point3D(0, 0, 0)
Dim point1 As New Point3D(5, 0, 0)
Dim point2 As New Point3D(0, 0, 5)
triangleMesh.Positions.Add(point0)
triangleMesh.Positions.Add(point1)
triangleMesh.Positions.Add(point2)
triangleMesh.TriangleIndices.Add(0)
triangleMesh.TriangleIndices.Add(2)
triangleMesh.TriangleIndices.Add(1)
Dim normal As New Vector3D(0, 1, 0)
triangleMesh.Normals.Add(normal)
triangleMesh.Normals.Add(normal)
triangleMesh.Normals.Add(normal)
Dim material As Material = New DiffuseMaterial(New
SolidColorBrush(Colors.DarkKhaki))
Dim triangleModel As New GeometryModel3D(triangleMesh, material)
Dim model As New ModelVisual3D()
model.Content = triangleModel
Me.mainViewport.Children.Add(model)
```

والناتج سيكون:

الصورة 16. 22. نتائج التنفيذ.

اما لرسم المكعب:

C# كود

```
private Model3DGroup CreateTriangleModel(Point3D p0, Point3D p1, Point3D p2){
    MeshGeometry3D mesh = new MeshGeometry3D();
    mesh.Positions.Add(p0);    mesh.Positions.Add(p1);  mesh.Positions.Add(p2);
    mesh.TriangleIndices.Add(0);
    mesh.TriangleIndices.Add(1);
    mesh.TriangleIndices.Add(2);
    Vector3D normal = CalculateNormal(p0, p1, p2);
    mesh.Normals.Add(normal);
    mesh.Normals.Add(normal);
    mesh.Normals.Add(normal);
    Material material = new DiffuseMaterial(new
SolidColorBrush(Colors.DarkKhaki));
    GeometryModel3D model = new GeometryModel3D(mesh, material);
    Model3DGroup group = new Model3DGroup();
    group.Children.Add(model);
    return group;
}
private Vector3D CalculateNormal(Point3D p0, Point3D p1, Point3D p2){
    Vector3D v0 = new Vector3D(p1.X - p0.X, p1.Y - p0.Y, p1.Z - p0.Z);
    Vector3D v1 = new Vector3D(p2.X - p1.X, p2.Y - p1.Y, p2.Z - p1.Z);
    return Vector3D.CrossProduct(v0, v1);
}
```

VB.NET كود

```
Private Function CreateTriangleModel(ByVal p0 As Point3D, ByVal p1 As Point3D,
ByVal p2 As Point3D) As Model3DGroup
    Dim mesh As New MeshGeometry3D()
    mesh.Positions.Add(p0)
    mesh.Positions.Add(p1)
    mesh.Positions.Add(p2)
    mesh.TriangleIndices.Add(0)
    mesh.TriangleIndices.Add(1)
    mesh.TriangleIndices.Add(2)
    Dim normal As Vector3D = CalculateNormal(p0, p1, p2)
    mesh.Normals.Add(normal)
    mesh.Normals.Add(normal)
    mesh.Normals.Add(normal)
    Dim material As Material = New DiffuseMaterial(New
SolidColorBrush(Colors.DarkKhaki))
    Dim model As New GeometryModel3D(mesh, material)
    Dim group As New Model3DGroup()
    group.Children.Add(model)
    Return group
End Function
Private Function CalculateNormal(ByVal p0 As Point3D, ByVal p1 As Point3D,
ByVal p2 As Point3D) As Vector3D
    Dim v0 As New Vector3D(p1.X - p0.X, p1.Y - p0.Y, p1.Z - p0.Z)
    Dim v1 As New Vector3D(p2.X - p1.X, p2.Y - p1.Y, p2.Z - p1.Z)
    Return Vector3D.CrossProduct(v0, v1)
End Function
```

وفي زر الأمر:

C# كود

```
MeshGeometry3D triangleMesh = new MeshGeometry3D();
Point3D point0 = new Point3D(0, 0, 0);
Point3D point1 = new Point3D(5, 0, 0);
Point3D point2 = new Point3D(0, 0, 5);
triangleMesh.Positions.Add(point0);
triangleMesh.Positions.Add(point1);
triangleMesh.Positions.Add(point2);
triangleMesh.TriangleIndices.Add(0);
triangleMesh.TriangleIndices.Add(2);
triangleMesh.TriangleIndices.Add(1);
Vector3D normal = new Vector3D(0, 1, 0);
triangleMesh.Normals.Add(normal);
triangleMesh.Normals.Add(normal);
triangleMesh.Normals.Add(normal);
Material material = new DiffuseMaterial(
    new SolidColorBrush(Colors.DarkKhaki));
GeometryModel3D triangleModel = new GeometryModel3D(
    triangleMesh, material);
ModelVisual3D model = new ModelVisual3D();
model.Content = triangleModel;
this.mainViewport.Children.Add(model);
```

VB.NET كود

```
Dim triangleMesh As New MeshGeometry3D()
Dim point0 As New Point3D(0, 0, 0)
Dim point1 As New Point3D(5, 0, 0)
Dim point2 As New Point3D(0, 0, 5)
triangleMesh.Positions.Add(point0)
triangleMesh.Positions.Add(point1)
triangleMesh.Positions.Add(point2)
triangleMesh.TriangleIndices.Add(0)
triangleMesh.TriangleIndices.Add(2)
triangleMesh.TriangleIndices.Add(1)
Dim normal As New Vector3D(0, 1, 0)
triangleMesh.Normals.Add(normal)
triangleMesh.Normals.Add(normal)
triangleMesh.Normals.Add(normal)
Dim material As Material = New DiffuseMaterial(New
SolidColorBrush(Colors.DarkKhaki))
Dim triangleModel As New GeometryModel3D(triangleMesh, material)
Dim model As New ModelVisual3D()
model.Content = triangleModel
Me.mainViewport.Children.Add(model)
```

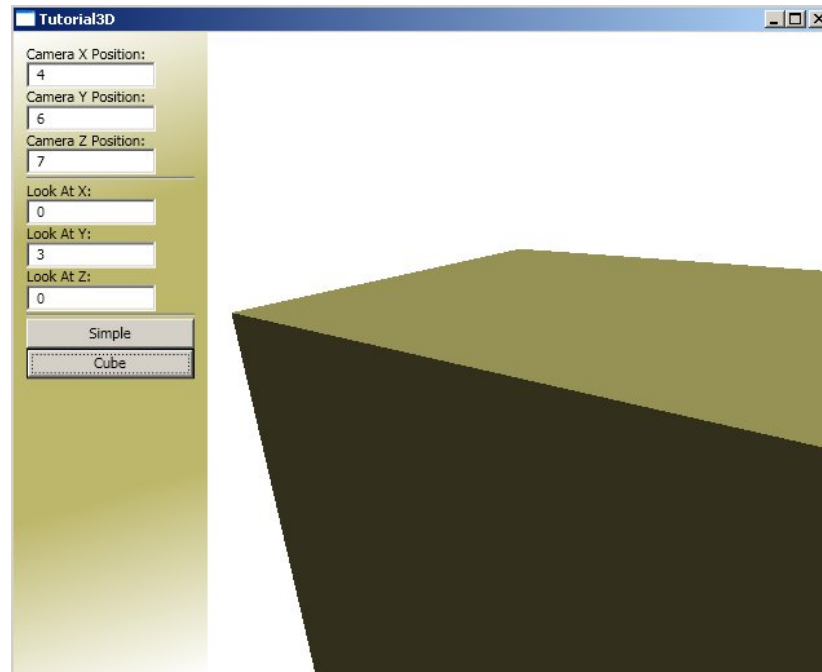
والناتج:

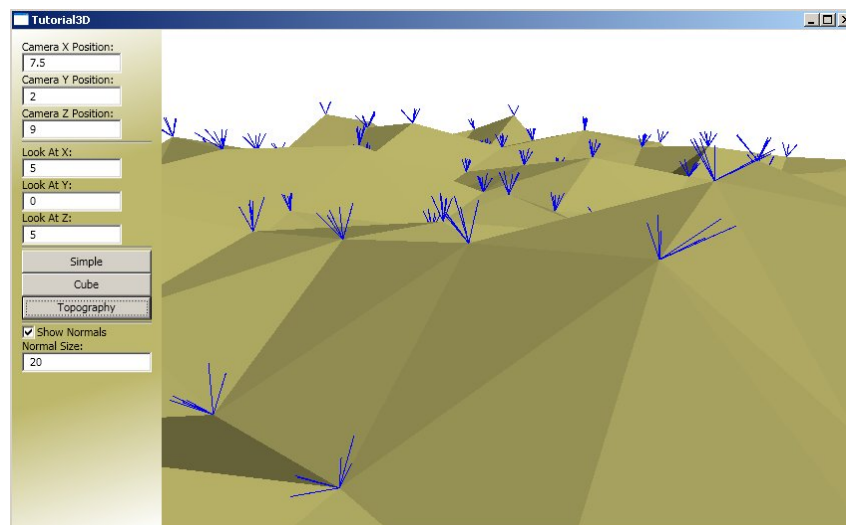
الصورة 16. 23. نتائج التنفيذ.

يمكنك المواصلة في هذه التطبيقات من المصدر الذي استقيت منه محتويات هذا الدرس على الرابط التالي:

رابط

http://www.kindohm.com/technical/wpf3dtutorial.htm

ويمكنك المتابعة للاطلاع على كيفية عمل برنامج يمكنك من التحكم في الكاميرا ليكون الناتج بالشكل التالي:

الصورة 16. 24. التحكم في وضعية الكاميرا و تغيير الزوايا.

او رسم ارضيات لتكون بالشكل التالي:

الصورة 16. 25. التحكم في وضعية الكاميرا و تغيير الزوايا.

14. عالم XNA

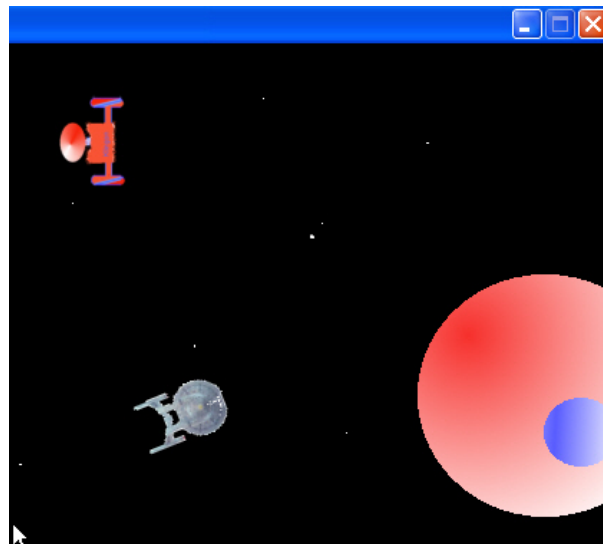
إذا كنت قد جربت التعامل مع برمجة الألعاب أو أي شيء له علاقة بالعالم الثلاثي الأبعاد، فأنت قد جربت التعامل مع Direct3D أو مع OpenGL أو غيرهم من المكتبات الخاصة بالتعامل مع العالم الثلاثي الأبعاد ، لذا لن يكون التعريف صعباً لك لو قلت لك أن ال XNA هي مجموعة جديدة من ال API's مبنية على DirectX تهدف إلى تسهيل التعامل مع مكتبات ال DirectX حيث تم تجهيز عدد كبير من الدوال والمهام لتنفيذ بعض العمليات التي كان تنفيذها يأخذ الكثير من الجهد.

ال XNA هي اختصار للجملة: DirectX Next Generation Architecture ، وهي مجانية ولكنها تشترط نسخة Express من الفيجوال ستوديو لتعمل ، لاحقاً سيكون بامكانك انشاء مشروع XNA بسهولة ، قم بتحميل XNA من الرابط التالي:

لو كنت تحاول برمجة لعبة بسيطة يمكنك الاستفادة من هذا الرابط:

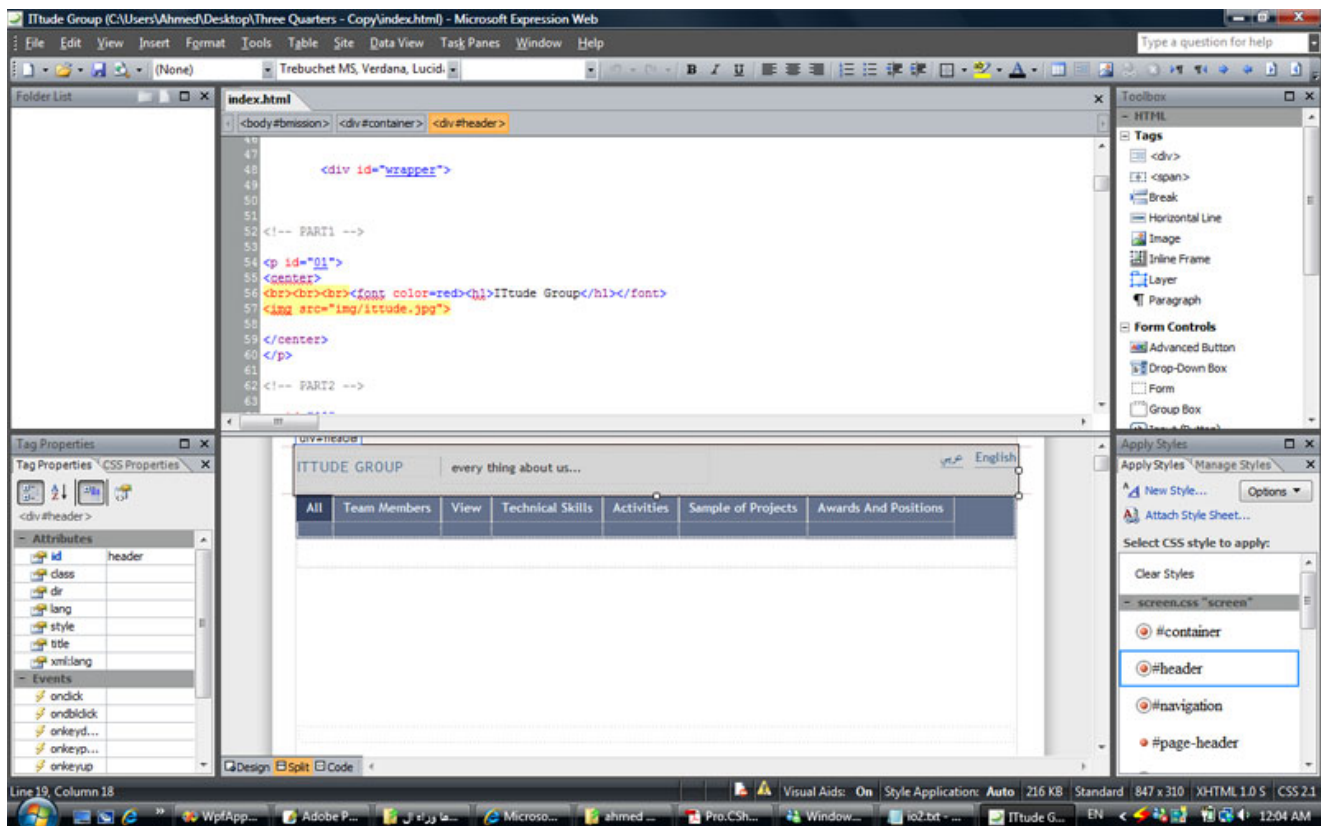
رابط

http://www.c-sharpcorner.com/UploadFile/mgold/XNAIntro04192007233237PM/XNAIntro.aspx

وسيكون ناتج تجربتك بالشكل التالي:

الآن إذا كنت مبرمج DirectX سابق فأنت بالطبع ادركت الفارق ... أما لو لم تكن كذلك فحاول البرمجة باستخدام DirectX لفترة ثم جرب الفارق...

15. Microsoft Expression Studio

15. 1. Microsoft Expression Web

يتيح لك هذا البرنامج تصميم صفحاتك بطريقة مميزة ويساعدك على عمل صفحتك بأي صيغة موجودة مثل XHTML, CSS, XML, XSLT . الواجهة الرئيسية له بالشكل التالي:

الصورة 16. 26. الواجهة الرئيسية لبرنامج ال Expression Web.

يمكنك استخدام اي من الادوات التي تجدها لديك لتصميم صفحتك والتعديل على الخصائص ، هناك ايضاً مدقق HTML لمراقبة الأخطاء، هناك خصائص لتسهيل التعامل مع ال styles مثلاً ، ليس هذا فقط بل يتيح لك البرنامج البرمجة من خلاله سواء من خلال Asp.net أو من خلال php.

للمزيد يمكنك البدء من هنا:

رابط

http://www.microsoft.com/expression/features/default.aspx?key=web

15 .2. Microsoft Expression Design

يقدم لك هذا البرنامج حلولاً مميزة لتسهيل تركيب الصور والرسوميات وخلافه ، الواجهة الرئيسية له بالشكل التالي:

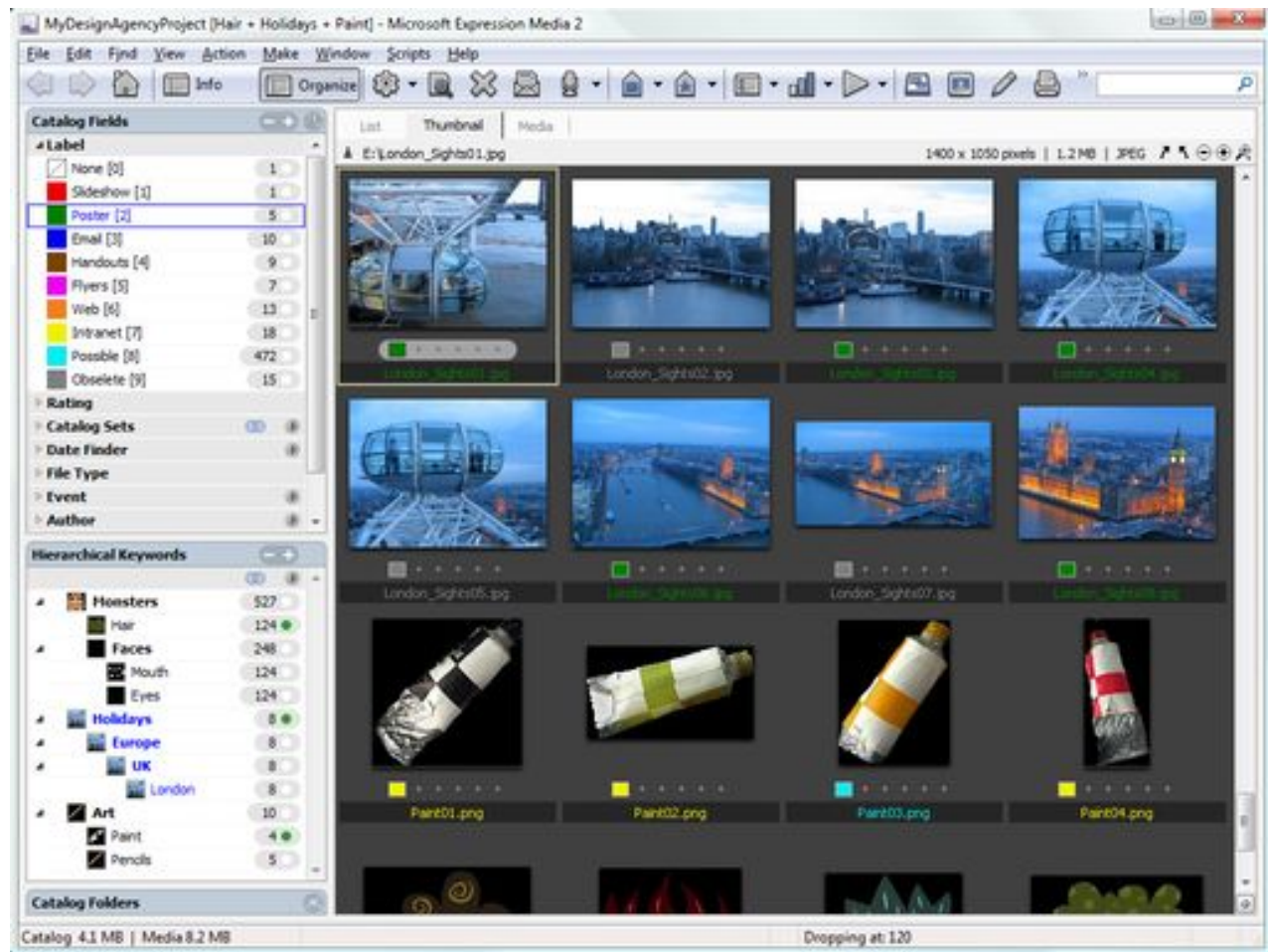
الصورة 16. 27. الواجهة الرئيسية لبرنامج ال Expression Design.

المزيد يمكنك معرفته من هنا:

رابط

http://www.microsoft.com/expression/features/default.aspx?key=design

15. 3. Microsoft Expression Media

يتيح لك هذا البرنامج تعديل وتركيب وتكوين أفلامك المختلفة ، هذه صورة للواجهة:

الصورة 16. 28. الواجهة الرئيسية لبرنامج ال Expression Design.

والمزيد يمكنك معرفته من هنا:

رابط

http://www.microsoft.com/expression/features/default.aspx?key=media

15. 4. Microsoft Expression Encoder

يتيح لك السماح للزائر بالتفاعل مع عروضك ، يشكل البرنامج الأساسي لادارة Silver Light التقنية الجديدة من مايكروسوفت والمشابهة لتقنية Flash من Macromedia سابقاً ومن Adobe حالياً.

الواجهة الرئيسية للبرنامج:

الصورة 16. 29. الواجهة الرئيسية لبرنامج ال Expression Encoder.

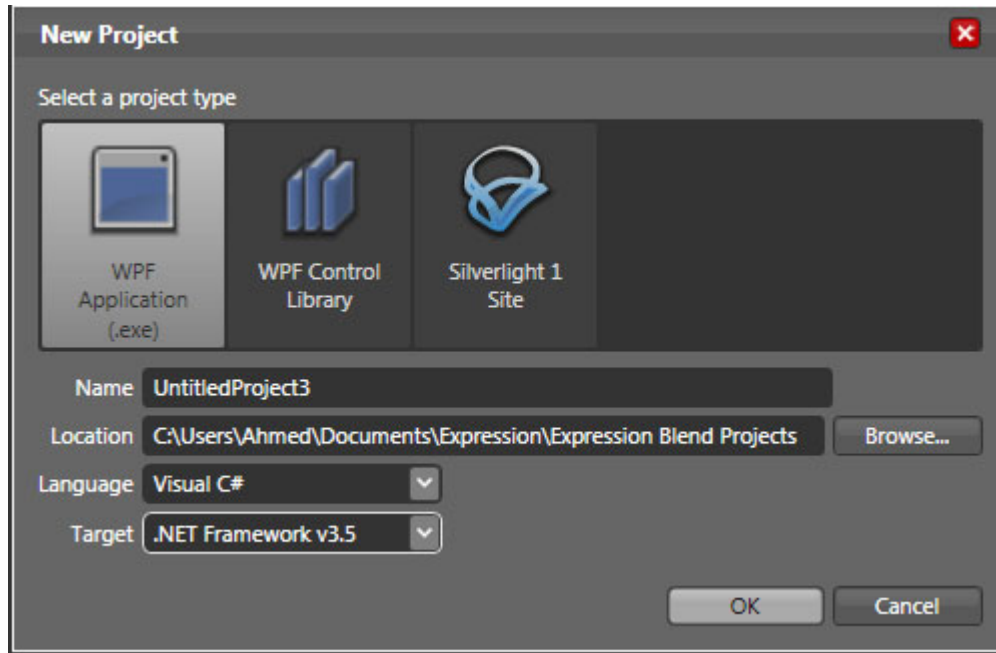
يمكنك البدء بمزيد من التفصيل هنا:

رابط

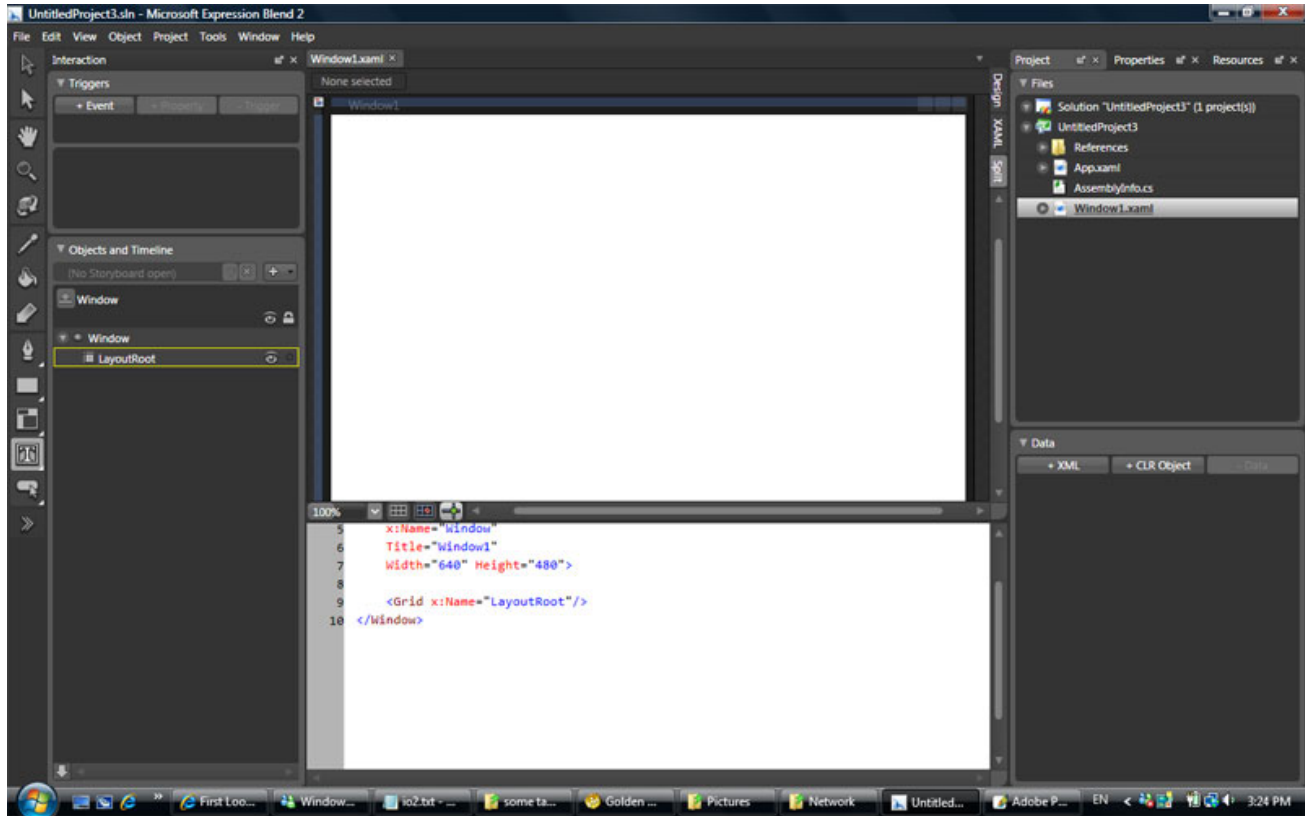
http://www.microsoft.com/expression/features/default.aspx?key=encoder

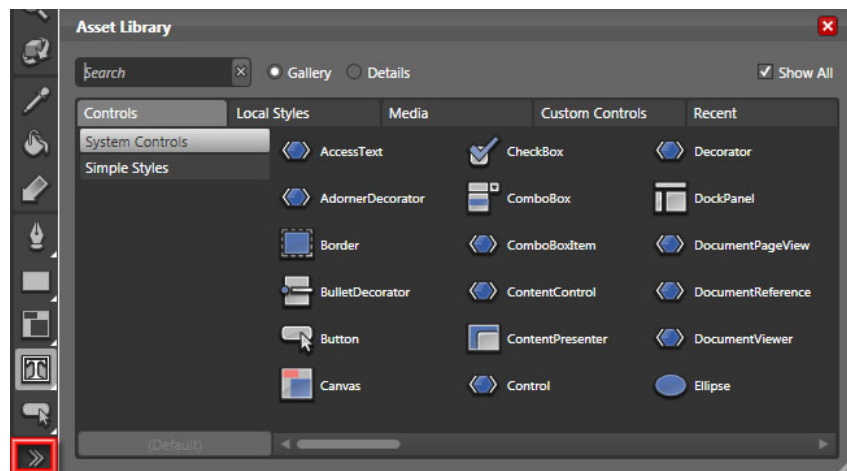
15. 5. Microsoft Expression Blend

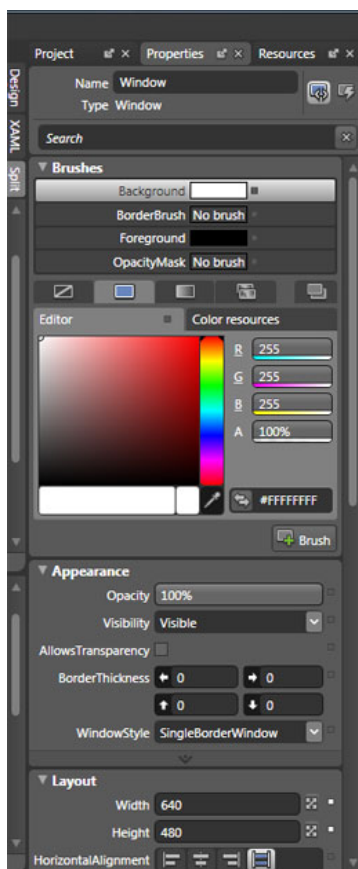
لتصميم واجهات برنامجك المختلفة، حيث يوفر لك وسائل متعددة للتصميم، سنحاول معرفة المزيد عنه بتفصيل في هذا الدرس...

سنحاول الآن عمل تطبيق بسيط من خلال Expression Blend قم بتحميله أولاً من الروابط السابقة، قم بتشغيله ومن ثم اختيار New Project ومن ثم WPF Application (*.exe) بالشكل التالي:

الصورة 16. 29. بناء مشروع جديد في ال Expression Blend

يمكنك اختيار اللغة التي تريد العمل عليها وال Framework اضافة للاسم ومكان التخزين بالطبع.

من View اختر Active Document View ومن ثم اختر الوضع Split لتتمكن من عرض XAML و العرض العادي في نفس الوقت بالشكل التالي:

الصورة 16. 30. الواجهة الرئيسية لبرنامج ال Expression Blend

على جانب الشاشة ، ستجد كل الأدوات التي تحتاج إليها للتصميم سواء القلم والفرشاة وخلافه ، أو أزرار الأوامر ومربعات النصوص وخلافه من الأدوات التي ستستخدمها في برنامجك ، ايضاً يمكنك الضغط على آخر عناصر القائمة ليستعرض لك جميع الأدوات بالشكل التالي مثلاً:

الصورة 16. 31. الواجهة الرئيسية لبرنامج ال Expression Blend

خلال عملياتك في الرسم ، ستجد لأي اداة مجموعة من الخصائص على الجانب ، تستطيع منها التحكم بالمظهر و خلافه:

الصورة 16. 32. نافذة الخصائص ال Expression Blend

الآن كتطبيق سريع ، قم باضافة Canvas وقم بتحديد ابعاده ، هذا هو الCanvas الذي سنضع فيه صورة خاصة بنا ونطبق عليها بعض التأثيرات:

كود	XAML

```
        <Canvas>
            <Canvas x:Name="MainImageCanvas" Canvas.Left="40" Canvas.Top="120">
```

الآن قم باضافة MediaElement ولنضع فيه صورة مثلاً – أو قم بوضع صورة مباشرة - ، سيكون ناتج XAML بالشكل التالي:

كود	XAML

```
<MediaElement x:Name="MainImage" Source="c:/example/futex.jpg" Width="300"
Height="300" >
</MediaElement>
```

والآن سنقوم بتطبيق بعض التأثيرات على ال Canvas حيث نطبق عملية الميل من خلال الخصائص ، سيكون ناتج XAML بالشكل التالي:

كود XAML

```
    <Canvas.RenderTransform>
        <TransformGroup>
            <SkewTransform x:Name="MainSkewTransform" AngleY="-19" AngleX="0"
CenterX="0" CenterY="0"/>
            <ScaleTransform x:Name="MainScaleTransform" ScaleY="1" ScaleX = "1"
CenterX="0" CenterY="0"/>
        </TransformGroup>
    </Canvas.RenderTransform>
</Canvas>
```

الآن قم بالضغط على F5 لتجربة العرض والذي سيكون بالشكل التالي:

الصورة 16. 33. نتائج التنفيذ.

سنحاول تطبيق نظرية الظل للصورة أيضاً ، يمكنك تطبيقها مباشرة من خصائص الفرشاة، أو في تجربتنا هنا سنضع صورتين على بعضهما ، لذا سنقوم بعمل Canvas ونضع فيه الصورة أيضاً ولكن مع زوايا ميل مختلفة هذه المرة بحيث تحاذي اطراف الصورة ، سيكون ناتج XAML بالشكل التالي:

كود XAML

```
<Canvas x:Name="ReflectionImageCanvas" Canvas.Left="260" Canvas.Top="640">
    <MediaElement x:Name="ReflImage" Source="c:/example/futex.jpg" Width="300"
Height="300" Volume="0">
    </MediaElement>
    <Canvas.RenderTransform>
        <TransformGroup>
            <SkewTransform x:Name="ReflectionSkewTransform" AngleY="19"
AngleX="-41" CenterX="0" CenterY="0" />
            <ScaleTransform x:Name="ReflectionScaleTransform" ScaleY="-1"
ScaleX="1" CenterX="0" CenterY="0" />
        </TransformGroup>
    </Canvas.RenderTransform>
</Canvas>
```

وسيكون الناتج للصورة بالشكل التالي:

الصورة 16. 34. نتائج التنفيذ.

آخر نقطة سنتعامل معها هي اضافة الشفافية لصورة الظل ، من ضمن الخصائص أيضاً ، لذا ستجد ناتج ال XAML في النهاية بالشكل التالي:

كود XAML

```
<Canvas x:Name="ReflectionImageCanvas" Canvas.Left="260" Canvas.Top="640">
    <MediaElement x:Name="ReflImage" Source="c:/example/futex.jpg" Width="300"
Height="300" Volume="0">
    </MediaElement>
    <Canvas.RenderTransform>
        <TransformGroup>
            <SkewTransform x:Name="ReflectionSkewTransform" AngleY="19"
AngleX="-41" CenterX="0" CenterY="0" />
            <ScaleTransform x:Name="ReflectionScaleTransform" ScaleY="-1"
ScaleX="1" CenterX="0" CenterY="0" />
        </TransformGroup>
    </Canvas.RenderTransform>
    <Canvas.OpacityMask>
        <LinearGradientBrush StartPoint="0.5,0.0" EndPoint="0.5,1.0">
            <GradientStop Offset="0.345" Color="#00000000"
x:Name="ReflGradientStop1" />
            <GradientStop Offset="1.0" Color="#CC000000"
x:Name="ReflGradientStop2" />
        </LinearGradientBrush>
    </Canvas.OpacityMask>
</Canvas>
```

قم بضبط بعض اعدادات ال Left وال Top يدوياً او من الكود لجعل صورة الظل منطبقة على الصورة الاصلية ، سيكون ناتج الصورة:

الصورة 16. 35. نتائج التنفيذ.

الكود الكامل XAML:

كود XAML

```
<Canvas>
    <Canvas x:Name="MainImageCanvas" Canvas.Left="40" Canvas.Top="120">

        <MediaElement x:Name="MainImage" Source="c:/example/FUTEX.JPG"
Width="300" Height="300" >
        </MediaElement>
        <Canvas.RenderTransform>
            <TransformGroup>
                <SkewTransform x:Name="MainSkewTransform" AngleY="-19"
AngleX="0" CenterX="0" CenterY="0"/>
                <ScaleTransform x:Name="MainScaleTransform" ScaleY="1" ScaleX =
"1" CenterX="0" CenterY="0"/>
            </TransformGroup>
        </Canvas.RenderTransform>
    </Canvas>
    <Canvas x:Name="ReflectionImageCanvas" Canvas.Left="267" Canvas.Top="645">
        <MediaElement x:Name="ReflImage" Source="c:/example/futex.jpg"
Width="300" Height="300" Volume="0">
        </MediaElement>
        <Canvas.RenderTransform>
            <TransformGroup>
                <SkewTransform x:Name="ReflectionSkewTransform" AngleY="19"
AngleX="-41" CenterX="0" CenterY="0" />
                <ScaleTransform x:Name="ReflectionScaleTransform" ScaleY="-1"
ScaleX="1" CenterX="0" CenterY="0" />
            </TransformGroup>
        </Canvas.RenderTransform>
        <Canvas.OpacityMask>
            <LinearGradientBrush StartPoint="0.5,0.0" EndPoint="0.5,1.0">
                <GradientStop Offset="0.345" Color="#00000000"
x:Name="ReflGradientStop1" />
                <GradientStop Offset="1.0" Color="#CC000000"
x:Name="ReflGradientStop2" />
            </LinearGradientBrush>
        </Canvas.OpacityMask>
    </Canvas>
</Canvas>
```

لا تنس ان MediaElement يمكن ان تكون اي شيء، لذا جرب مثلاً وضع فيديو وستجد ان نفس التأثير ينطبق عليه تماماً...

طبعاً يمكنك نقل الكود كما هو إلى الفيجوال ستوديو وسيعمل بنفس الصورة.

قبل النهاية ، احب ان اذكرك بأنك كمبرمج لست مطالباً بمعرفة الكثير عن هذا العالم عالم Microsoft Expression فهو موجه اصلاً للمصممين ، كل ما يهمك هو التعامل مع ال XAML الناتجة فقط.

لتعلم كل ما تريد عن هذه المجموعة يمكنك البدء من هنا:

رابط

http://expression.microsoft.com/en-us/cc136522.aspx

الباب

17

قواعد البيانات

ADO.net

مقدمة

ال Active Data Object.Net او ما تعرف اختصاراً باسم ADO.net هي امتداد للنسخة القديمةADO التي كانت موجودة في اصدارات فيجوال بيسك 6 من مايكروسوفت ، وهي امتداد ايضاً لعالم مزودات البيانات او Data Providers التي مرت بمراحل تطور عديدة وشهدت العديد من التقنيات مثلDAO, RDO .

وعلى عكس النسخة القديمة ADO التي كانت مخصصة لفتح اتصال بين قاعدة البيانات والبرنامج او بين الخادم والعميل Client/Server فإن ADO.net اصبحت مزودة بخدمة Disconnected بحيث لن تصبح مضطراً للاتصال دائماً بقاعدة البيانات .

هناك فروقات أخرى بين التقنيتين هذا موجزها:

- الدعم الكامل لXML .

- زيادة أنواع البيانات المعتمدة والدوال المستخدمة.

- زيادة السرعة.

- انها اصبحت managed code بالكامل.

يمكنك الاطلاع على مزيد من التفاصيل حول الفروقات هنا:

رابط

http://msdn.microsoft.com/en-us/library/904fck4k(VS.71).aspx

1. البدأ باستخدام ال ADO.net

فقط كل ما عليك هو استيراد المكتبات الخاصة بها ، تجدها جميعاً تحت اسم System.Data ، بعدها سيكون عليك تقرير فيما إذا كنت ترغب في العمل من خلال الوضع Connected أو Disconnected ، في الحالة الأولى ستكون على اتصال دائم بقاعدة البيانات اما الحالة الثانية فستتصل على مراحل لتجلب البيانات او لتقوم ببعض التعديلات الجماعية ، في بداية دروسنا لن نتعامل سوى مع Connected فيما سنعود لاحقاً للوضع الآخر.

2. مكونات ال ADO.net Data Provider

1- Connection : يمكنك من انشاء اتصالك بقاعدة البيانات.
2- Command : تمكنك من تنفيذ جملة استعلام SQL على قاعدة البيانات.
3- IDataReader DbDataReader DataReader: تمكنك من قراءة البيانات الناتجة عن عملية استعلام على سبيل المثال ، لكن الناتج سيكون للقراءة فقط وفي اتجاه واحد فقط.
4- IDataAdapter DbDataAdapter DataAdapter : تمكنك على الحصول على صورة من البيانات المطلوبة سواء جدول او جملة استعلام وتمكنك من التنقل بينها وعمل اضافة وتعديل وحذف وخلافه.

5- IDataParameter DbParameter Parameter : خاصة بتمرير متغيرات إلى جمل الاستعلام.

2. 1. ال Data Providers المدعومة من قبل مايكروسوفت

System.Data.OleDb:

يمكن استخدامها للتعامل مع اي نوع من قواعد البيانات ، لكن لو كان لقاعدة البيانات مزود آخر موجود في القائمة يفضل استخدامه نظراً لإن هذا النوع هو الأبطأ.

System.Data.SqlClient:

يفضل دوماً استخدامها للتعامل مع قواعد البيانات SQL Server حيث انها تحتوي على مجموعة من المهام الخاصة بالتعامل مع هذا النوع من قواعد البيانات.

:System.Data.SqlServerCe

يفضل استخدامها في حالة التعامل مع قاعدة بيانات SQL Server CE الخاصة بالتعامل مع ال Pocket PC.

:System.Data.Odbc

اي ملف Odbc يمكن التعامل معه من خلالها.

:System.Data.OracleClient

يفضل التعامل مع قواعد البيانات أوراكل من خلالها.

2 .2. التعامل مع مزودات خدمة أخرى Third-Party ADO.NET Data Providers

في الفقرة السابقة ذكرنا مزودات لقواعد البيانات الرئيسية، ولكن ماذا عن قواعد البيانات MYSQL أو FoxPro أو DB2 مثلاً ؟

هل سنضطر في النهاية لاستخدام OleDb ذات السرعة الأبطأ والامكانيات المحدودة نسبياً ؟

لا طبعاً ، قامت اغلب الشركات بتطوير Data Providers لمنتجاتها خاصة ب ADO.net أو حتى شركات تقدم مزودات معتمدة من الشركات الأصلية تتمتع بسرعة اكبر ومميزات وخدمات اضافية ، هذا الرابط مثال عليها:

رابط

http://www.sqlsummit.com/dataprov.htm

3. مكونات مجال الأسماء System.Data

يحتوي هذه الفئة على كل ما يخص قواعد البيانات والتعامل معها من دوال وخصائص وحتى رسائل الأخطاء Exceptions، سنستعرض هنا سريعاً أهم محتويات هذه الفئة حيث ستفيدنا في التعرف على خصائصها:

IDbConnection:

منه يتم اشتقاق الفئة الخاصة بالاتصال بقاعدة البيانات، يحتوي على الدوال التالية:

كود — C#

```
public interface IDbConnection : IDisposable
{
    string ConnectionString { get; set; }
    int ConnectionTimeout { get; }
    string Database { get; }
    ConnectionState State { get; }
    IDbTransaction BeginTransaction();
    IDbTransaction BeginTransaction(IsolationLevel il);
    void ChangeDatabase(string databaseName);
    void Close();
    IDbCommand CreateCommand();
    void Open();
}
```

كود — VB.NET

```
Public Interface IDbConnection
    Inherits IDisposable
    Property ConnectionString() As String
    ReadOnly Property ConnectionTimeout() As Integer
    ReadOnly Property Database() As String
    ReadOnly Property State() As ConnectionState
    Function BeginTransaction() As IDbTransaction
    Function BeginTransaction(ByVal il As IsolationLevel) As IDbTransaction
    Sub ChangeDatabase(ByVal databaseName As String)
    Sub Close()
    Function CreateCommand() As IDbCommand
    Sub Open()
End Interface
```

IDbCommand:

يتم منه اشتقاق الفئات الخاصة بالتعامل لاحقاً مع تنفيذ جمل الاستعلام ، يحتوي على الدوال التالية:

كود | C#

```
public interface IDbCommand : IDisposable
{
    string CommandText { get; set; }
    int CommandTimeout { get; set; }
    CommandType CommandType { get; set; }
    IDbConnection Connection { get; set; }
    IDataParameterCollection Parameters { get; }
    IDbTransaction Transaction { get; set; }
    UpdateRowSource UpdatedRowSource { get; set; }
    void Cancel();
    IDbDataParameter CreateParameter();
    int ExecuteNonQuery();
    IDataReader ExecuteReader();
    IDataReader ExecuteReader(CommandBehavior behavior);
    object ExecuteScalar();
    void Prepare();
}
```

كود | VB.NET

```
Public Interface IDbCommand
    Inherits IDisposable
    Property CommandText() As String
    Property CommandTimeout() As Integer
    Property CommandType() As CommandType
    Property Connection() As IDbConnection
    ReadOnly Property Parameters() As IDataParameterCollection
    Property Transaction() As IDbTransaction
    Property UpdatedRowSource() As UpdateRowSource
    Sub Cancel()
    Function CreateParameter() As IDbDataParameter
    Function ExecuteNonQuery() As Integer
    Function ExecuteReader() As IDataReader
    Function ExecuteReader(ByVal behavior As CommandBehavior) As IDataReader
    Function ExecuteScalar() As Object
    Sub Prepare()
End Interface
```

IDbDataParameter :

تحتوي على:

كود | C#

```
public interface IDbDataParameter : IDataParameter
{
    byte Precision { get; set; }
    byte Scale { get; set; }
    int Size { get; set; }
}
```

كود VB.NET
```
Public Interface IDbDataParameter
    Inherits IDataParameter
    Property Precision() As Byte
    Property Scale() As Byte
    Property Size() As Integer
End Interface
```

IDataParameter :

تحتوي على:

كود C#
```
public interface IDataParameter
{
    DbType DbType { get; set; }
    ParameterDirection Direction { get; set; }
    bool IsNullable { get; }
    string ParameterName { get; set; }
    string SourceColumn { get; set; }
    DataRowVersion SourceVersion { get; set; }
    object Value { get; set; }
}
```

كود VB.NET
```
Public Interface IDataParameter
    Property DbType() As DbType
    Property Direction() As ParameterDirection
    ReadOnly Property IsNullable() As Boolean
    Property ParameterName() As String
    Property SourceColumn() As String
    Property SourceVersion() As DataRowVersion
    Property Value() As Object
End Interface
```

IDbDataAdapter :

كود C#
```
public interface IDbDataAdapter : IDataAdapter
{
    IDbCommand DeleteCommand { get; set; }
    IDbCommand InsertCommand { get; set; }
    IDbCommand SelectCommand { get; set; }
    IDbCommand UpdateCommand { get; set; }
}
```

VB.NET كود

```
Public Interface IDbDataAdapter
    Inherits IDataAdapter
    Property DeleteCommand() As IDbCommand
    Property InsertCommand() As IDbCommand
    Property SelectCommand() As IDbCommand
    Property UpdateCommand() As IDbCommand
End Interface
```

IDataAdapter :

C# كود

```
public interface IDataAdapter
{
    MissingMappingAction MissingMappingAction { get; set; }
    MissingSchemaAction MissingSchemaAction { get; set; }
    ITableMappingCollection TableMappings { get; }
    int Fill(System.Data.DataSet dataSet);
    DataTable[] FillSchema(DataSet dataSet, SchemaType schemaType);
    IDataParameter[] GetFillParameters();
    int Update(DataSet dataSet);
}
```

VB.NET كود

```
Public Interface IDataAdapter
    Property MissingMappingAction() As MissingMappingAction
    Property MissingSchemaAction() As MissingSchemaAction
    ReadOnly Property TableMappings() As ITableMappingCollection
    Function Fill(ByVal dataSet As System.Data.DataSet) As Integer
    Function FillSchema(ByVal dataSet As DataSet, ByVal schemaType As
SchemaType) As DataTable()
    Function GetFillParameters() As IDataParameter()
    Function Update(ByVal dataSet As DataSet) As Integer
End Interface
```

IDataReader :

C# كود

```
public interface IDataReader : IDisposable, IDataRecord
{
    int Depth { get; }
    bool IsClosed { get; }
    int RecordsAffected { get; }
    void Close();
    DataTable GetSchemaTable();
    bool NextResult();
    bool Read();
}
```

VB.NET	كود

```
Public Interface IDataReader
    Inherits IDisposable
    Inherits IDataRecord
    ReadOnly Property Depth() As Integer
    ReadOnly Property IsClosed() As Boolean
    ReadOnly Property RecordsAffected() As Integer
    Sub Close()
    Function GetSchemaTable() As DataTable
    Function NextResult() As Boolean
    Function Read() As Boolean
End Interface
```

IDataRecord :

C#	كود

```
public interface IDataRecord
{
    int FieldCount { get; }
    object this[string name] { get; }
    object this[int i] { get; }
    bool GetBoolean(int i);
    byte GetByte(int i);
    char GetChar(int i);
    DateTime GetDateTime(int i);
    Decimal GetDecimal(int i);
    float GetFloat(int i);
    short GetInt16(int i);
    int GetInt32(int i);
    long GetInt64(int i);
    //...
    bool IsDBNull(int i);
}
```

كود	VB.NET

```
Public Interface IDataRecord
    ReadOnly Property FieldCount() As Integer
    Default ReadOnly Property Item(ByVal name As String) As Object
        Get
        End Get
    End Property
Default ReadOnly Property Item(ByVal i As Integer) As Object
    Get
    End Get
End Property
Function GetBoolean(ByVal i As Integer) As Boolean
    Function GetByte(ByVal i As Integer) As Byte
    Function GetChar(ByVal i As Integer) As Char
    Function GetDateTime(ByVal i As Integer) As DateTime
    Function GetDecimal(ByVal i As Integer) As Decimal
    Function GetFloat(ByVal i As Integer) As Single
    Function GetInt16(ByVal i As Integer) As Short
    Function GetInt32(ByVal i As Integer) As Integer
    Function GetInt64(ByVal i As Integer) As Long
    '...
    Function IsDBNull(ByVal i As Integer) As Boolean
End Interface
```

4. البداية مع SQL Server

4. 1. انشاء قاعدة البيانات

انتهينا من الفئات الاساسية في عالم ADO.net، وجاء وقت التطبيق الآن.

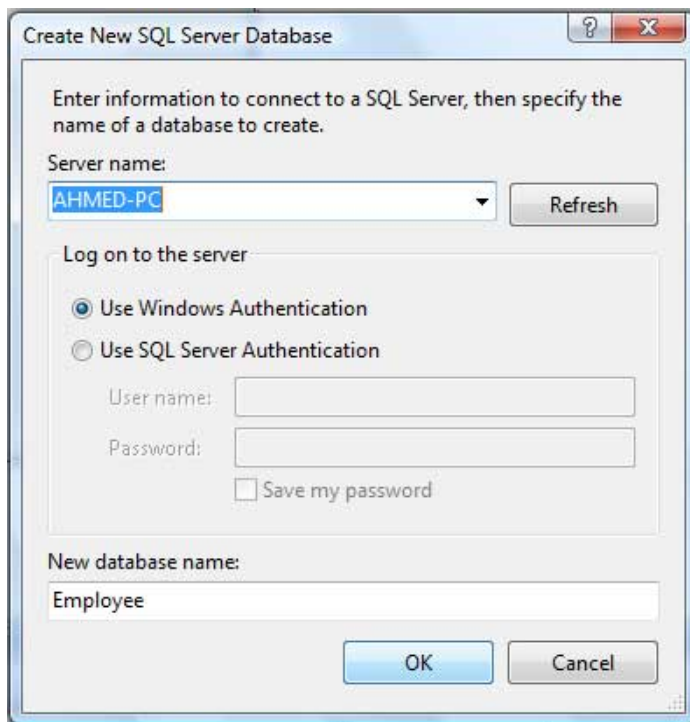
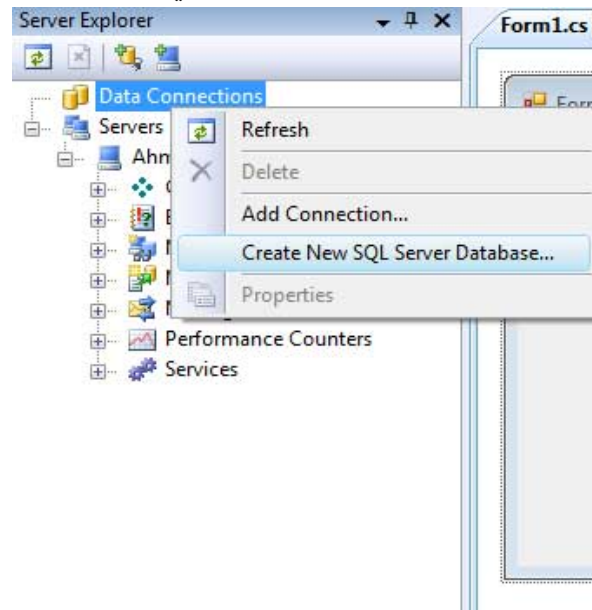
سنحاول الآن تطبيق انشاء قاعدة بيانات مثلاً للموظفين باسم Employee ، الآن قم بتشغيل نسخة فيجوال ستوديو جديدة Windows Forms، ومن ثم قم باختيار المسار والبيانات اللازمة.

بداية لا بد أن يكون لديك نسخة SQL Server من الاصدار السابع أو احدث او على الاقل النسخة الموجودة مع الفيجوال ستوديو Express على الرابط التالي :

رابط

http://msdn.microsoft.com/vstudio/express/sql

والتي تقع ضمن المجموعة المجانية من مايكروسوفت.

الآن من قائمة View قم باختيار Server Explorer ستجده على الجانب ، قم بالضغط بالزر الأيمن واختيار Create New Sql Server Database بالشكل التالي:

الصورة 17. 1. انشاء قاعدة بيانات جديدة من نوع SQL Server من نافذة ال Server Explorer في الفيجوال ستوديو.

الآن اصبح بامكانك انشاء قاعدة البيانات ، قم فقط بتحديد الاسم ثم اضغط Create بالشكل التالي

الصورة 17. 2. انشاء قاعدة بيانات جديدة من نوع SQL Server و تحديد معلوماتها.

*** قد تظهر لك مشكلة عدم وجود سيرفر في الاصل لتتصل به، في الغالب حل هذه المشكلة يكون لعدم اتصالك بالشبكة ، فقط قم بتوصيل الشبكة - حتى في حالة عدم وجود اتصال بالانترنت - وجرب . فقط .
*** قد يظهر لك اسم سيرفر ولكن تظهر لك رسالة الخطأ التالية :

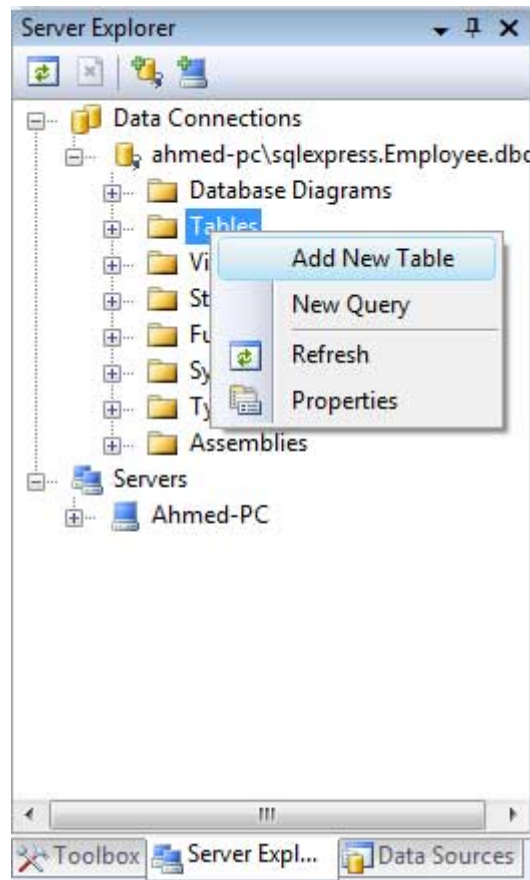
An error has occurred while establishing a connection to the server. When connecting to SQL Server 2005, this failure may be caused by the fact that under the default settings SQL Server does not allow remote connections. (provider: Named Pipes Provider, error: 40 - Could not open a connection to SQL Server)

في هذه الحالة من البرامج قم باختيار Microsoft SQL Server 2005 ومن ثم Configuration Tools ومن ثم SQL Server 2005 Surface Area Configuration ، ستجد من ضمن الخصائص Surface Area Configuration for Services and connections ، قم بفتحها والتأكد من ان خصائص Local and remote connections مفعلة ، قم ايضاً بالتأكد من أن السيرفر يعمل.

ايضاً يمكنك متابعة مجموعة من الحلول هنا :

رابط

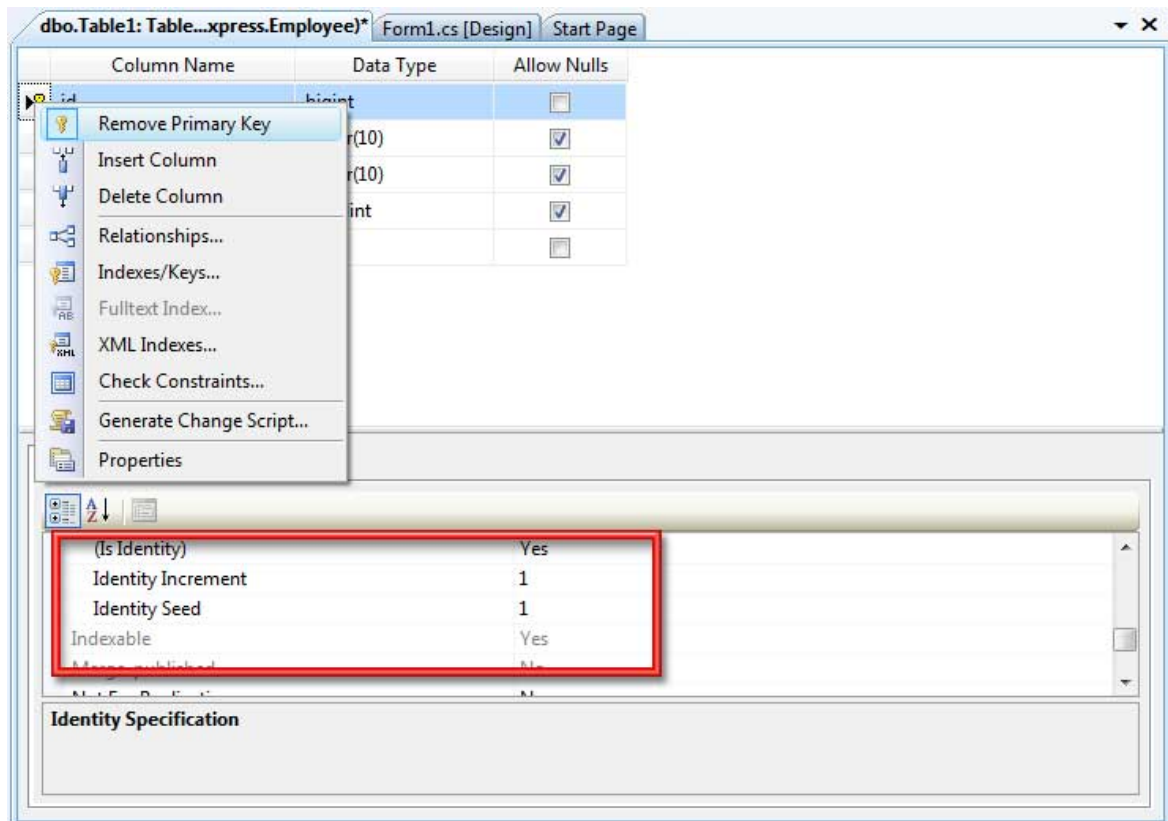
http://forums.microsoft.com/MSDN/ShowPost.aspx?PostID=192622&SiteID=1

الآن اصبح لديك قاعدة بيانات وجاهزة للعمل ، سنبدأ باضافة جدول بيانات الموظفين بالشكل التالي:

الصورة 17. 3. اضافة جدول إلى قاعدة البيانات.

لاحظ انه بامكانك اضافة اي محتويات بهذا الشكل مثل الدوال والاجراءاتStored Procedure وغيره ، الآن قمنا باختيار اضافة جدول بالشكل التالي:

Column Name	Data Type	Allow Nulls
id	bigint	☐
[First Name]	nchar(10)	☑
[last Name]	nchar(10)	☑
Age	smallint	☑
		☐

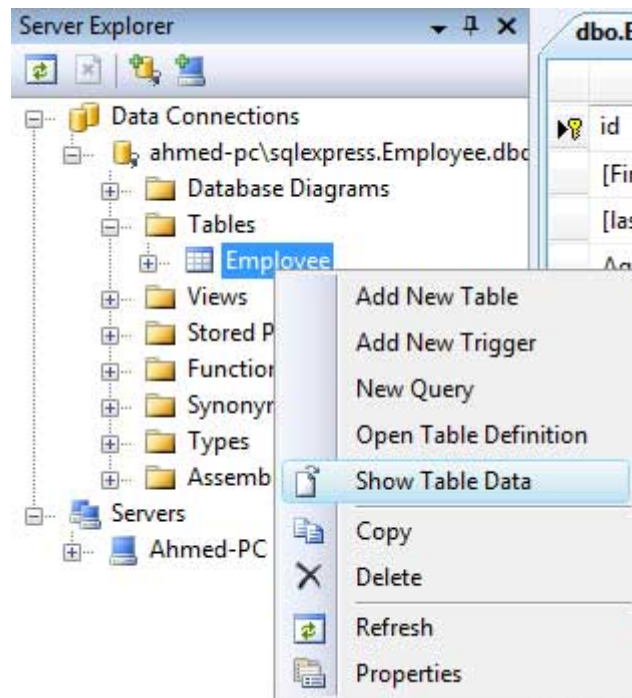
الصورة 17. 4. تعديل حقول الجدول.

لتعيين مفتاح رئيسي نقوم باختيار set primary key ولجعله autonumber نقوم بتعديل Identity Specfications إلى yes ومن ثم نقوم بتحديد Increment وهو معدل الزيادة إلى 1 كما في الشكل التالي:

الصورة 17. 5. اضافة المفتاح الرئيسي Primary Key لحقل من حقول الجدول.

وبنفس الطريقة يمكنك تعديل قيم وخصائص اي حقل ترغب فيه بنفس الطريقة.

سنتعرف على أنواع البيانات في الجزء اللاحق مباشرة .

اخيراً بعد الانتهاء قم بحفظ الجدول باسم Employee_Info مثلاً.

بعد ان قمت بالخطوات السابقة ، يمكنك استعراض الجداول التي لديك وفتحها للبدء في ادخال البيانات بالشكل التالي:

الصورة 17. 6. استعراض البيانات المخزنة في الجدول و التعديل و الاضافة عليها.

وبنفس الطريقة يمكنك اعادة تصميم الجدول وخلافه ، قمت بادخال بعض البيانات بالشكل التالي مثلاً:

Employee: Query(...express.Employee) | dbo.Employee: Ta...express.Employee)

	id	First Name	last Name	Age
	1	Ahmed	Gamal	22
	2	Hossam	Sadik	21
	3	Khaled	Adel	21
	4	Ahmed	Saied	21
	5	Ahmed	Essawy	23
	NULL	Ahmed	Emad	21
*	NULL	NULL	NULL	NULL

الصورة 17. 6. ادخال البيانات.

أنواع البيانات في SQL Server :

النوع	المجال من	المجال إلى
bigint	-9,223,372,036,854,775,808	9,223,372,036,854,775,807
int	-2,147,483,648	2,147,483,647
smallint	-32,768	32,767
tinyint	0	255
bit	0	1
decimal	-10^38 +1	10^38 –1
numeric	-10^38 +1	10^38 –1
money	-922,337,203,685,477.5808	+922,337,203,685,477.5807
smallmoney	-214,748.3648	+214,748.3647
float	-1.79E + 308	1.79E + 308
real	-3.40E + 38	3.40E + 38
datetime (3.33 milliseconds accuracy)	Jan 1, 1753	Dec 31, 9999
smalldatetime (1 minute accuracy)	Jan 1, 1900	Jun 6, 2079
char	حرف ثابت الحجم – يقبل حتى 8000 حرف	
varchar	حرف non-Unicode– يقبل حتى 8000 حرف	

varchar(max)	حروف non-Unicode تحمل حتى 231 حرف
text	حروف non-Unicode تحمل حتى 2,147,483,647 حرف
nchar	حروف ثابتة Unicode تحمل حتى 4,000 حرف
nvarchar	حروف Unicodeتحمل حتى 4,000 حرف
nvarchar(max)	حروف non-Unicode تحمل حتى 230 حرف
ntext	حروف Unicodeتحمل حتى 1,073,741,823 حرف.
binary	بيانات ثنائية ثابتة تحمل حتى 8,000 of بايت.
varbinary	بيانات ثنائية متغيرة تحمل حتى 8,000 of بايت.
varbinary(max)	بيانات ثنائية متغيرة تحمل حتى 231 بايت
image	بيانات ثنائية متغيرة تحمل حتى 2,147,483,647 بايت.

الجدول 17. 1. أنواع البيانات في ال SQL Server.

وإذا كنت قد تعودت على نظام تسمية أنواع البيانات في net. ، فهذا الجدول من مايكروسوفت يوضح لك نظير كل نوع بيانات في net. :

SQL SERVER	**نوع البيانات في net. CLR**	**نوع البيانات (SQL Server) CLR**
Bigint	Int64, Nullable<Int64>	SqlInt64
Binary	Byte[]	SqlBytes, SqlBinary

SqlBoolean	Boolean, Nullable<Boolean>	bit
لا يوجد	لا يوجد	char
لا يوجد	لا يوجد	cursor
SqlDateTime	DateTime, Nullable<DateTime>	date
SqlDateTime	DateTime, Nullable<DateTime>	datetime
SqlDateTime	DateTime, Nullable<DateTime>	datetime2
لا يوجد	DateTimeOffset, Nullable<DateTimeOffset>	DATETIMEOFFSET
SqlDecimal	Decimal, Nullable<Decimal>	decimal
SqlDouble	Double, Nullable<Double>	float
لا يوجد	لا يوجد	image
SqlInt32	Int32, Nullable<Int32>	int
SqlMoney	Decimal, Nullable<Decimal>	money
SqlChars, SqlString	String, Char[]	nchar
لا يوجد	لا يوجد	ntext
SqlDecimal	Decimal, Nullable<Decimal>	numeric
SqlChars, SqlString	String, Char[]	nvarchar
SqlChars, SqlString	Char, String, Char[], Nullable<char>	nvarchar(1), nchar(1)
SqlSingle	Single, Nullable<Single>	real
لا يوجد	Byte[]	rowversion
SqlInt16	Int16, Nullable<Int16>	smallint

SqlMoney	Decimal, Nullable<Decimal>	smallmoney
لا يوجد	Object	sql_variant
لا يوجد	لا يوجد	table
لا يوجد	لا يوجد	text
TimeSpan	TimeSpan, Nullable<TimeSpan>	time
لا يوجد	لا يوجد	timestamp
SqlByte	Byte, Nullable<Byte>	tinyint
SqlGuid	Guid, Nullable<Guid>	uniqueidentifier
لا يوجد	نفس الفئة المحددة	User-defined type(UDT)
SqlBytes, SqlBinary	Byte[]	varbinary
SqlBytes, SqlBinary	byte, Byte[], Nullable<byte>	varbinary(1), binary(1)
لا يوجد	لا يوجد	varchar
SqlXml	لا يوجد	xml

الجدول 17. 2. أنواع البيانات في ال SQL Server و مقارنتها مع أنواع البينات في الدوت نت.

4. 2. SQL Statements

لمعالجة البيانات في قاعدة البيانات وتنفيذ عمليات الاضافة والتعديل وخلافه اضافة لعمليات الإنشاء والتكوين نحتاج إلى لغة تفهمها قواعد البيانات ، هذه اللغة هي SQL وهي اختصار لـ Structured Query Language.

وتصلح تقنية الاستعلام للعمل مع جميع أنواع قواعد البيانات : Access - MS sql - Oracle Server وغيرها.

تقوم هذه الجملة بالبحث - كما هو شائع - إلا أنها تستخدم أيضاً في الاضافة والحذف والتعديل وانشاء الجداول وحتى قواعد البيانات . والتحكم بها والادارة وغير ذلك.

هذه هي الدوال الرئيسية في لغة الإستعلام هذه :

SELECT INSERT UPDATE DELETE MERGE	Data manipulation language (DML)
CREATE ALTER DROP RENAME TRUNCATE COMMENT	Data definition language (DDL)
COMMIT ROLLBACK SAVEPOINT	Transaction control
GRANT REVOKE	Data control language (DCL)

الصورة 17. 7. أنواع دوال الاستعلام في لغة ال SQL

الصيغة العامة لجملة الإستعلام الأشهر للبحث هي :

كود　　SQL

```
SELECT Colums_Name FROM Table_Name;
```

نلاحظ تكون جملة الإستعلام من أربعة أجزاء:

Select: وهي التي تميز جملة البحث عن غيرها من جمل SQL.

Colum_Name: اسم الحقل أو العمود الذي تريد له أن يظهر.

From: وهي أيضاً صيغة خاصة بجملة البحث .

Table_Name: هنا نكتب اسم الجدول.

ولنلق نظرة قبل البدء على ما يمكن تسميته (القواعد النحوية) Syntax لجمل الإستعلام :

* لا يوجد أي فرق بين الأحرف الكبيرة والصغيرة.
* مسافة = مسافتان = عشرة أسطر : لا يعترف بأكثر من مسافة.
* تنتهي جميع جمل الإستعلام بفاصلة منقوطة ; إلا أننا قد لا نحتاج إليها في تطبيقات البرمجة غالباً .

وهذا مثال على الجملة السابقة :

كود SQL
```sql
SELECT FName FROM Tb_Main;
```

وإذا كنا نريد استخراج أكثر من عمود فيمكننا وضعهم بالتوالي وتفصل بينهم الفاصلة ، فمثلاً لاستخراج الإسم الأول والإسم الأخير :

كود SQL
```sql
SELECT FName,LName FROM Tb_Main;
```

ولاستخراج جميع البيانات فإننا نضع * محل اسماء الحقول وذلك بالشكل التالي :

كود SQL
```sql
SELECT * FROM Tb_Main;
```

ولاستخراج البيانات ولكن بدون تكرار نستخدم ... DISTINCT فمثلاً لاستخراج الإسم الأول مع حذف التكرار :

كود SQL
```sql
SELECT DISTINCT FNAme FROM Tb_Main;
```

أيضاً لترتيب البيانات المخرجة تصاعدياً أو تنازلياً نستخدم الأمر:

كود SQL
```sql
... ORDER BY colum
```

أو للتنازلي

كود	SQL

```
... ORDER BY colum DESC
```

فمثلاً لاستخراج جمع البيانات مع الترتيب التصاعدي حسب الإسم الأول :

كود	SQL

```
SELECT * FROM Tb_Main ORDER BY FName DESC;
```

ويمكننا عمل مستويين للترتيب - كما هو شائع - وذلك باستخدام الفاصلة ، فمثلاً للترتيب حسب الإسم الأول ومن ثم الإسم الأخير ولكن تنازلياً هذه المرة :

كود	SQL

```
SELECT * FROM Tb_Main ORDER BY FName,Lname ;
```

تأخذ صيغة استخدام المساواة كشرط في عملية الإستعلام الصيغة التالية:

كود	SQL

```
SELECT * FROM Tb_Main WHERE Field = Value;
```

ويمكننا استخدام > أو < أو => أو =< أو <> وطبعاً=.

وأخيراً معاملين جدد سنشرحهم بإذن الله لاحقاً وهما LIKE وBETWEEN

فمثلاً للاستعلام عن الأصدقاء الذي تساوي أعمارهم 18 سنة :

كود	SQL

```
SELECT * FROM tb_Main WHERE age = 18;
```

لعرض أسماء من هم أصغر من 50 سنة:

كود	SQL

```
SELECT FName,LName FROM tb_Main WHERE age < 50;
```

وسنضع Value بين علامتي تنصيص مفردة في حالة كانت نصوصاً وذلك بالشكل التالي - للبحث عن الأشخاص والذين لهم الإسم (أحمد) :

كود	SQL

```
SELECT * FROM tb_Main WHERE Fname = 'ahmed';
```

كما سنضع علامتي # عند البحث عن تواريخ ، ولا توجد أمثلة لتاريخ في قاعدة البيانات ، لكنها تأخذ صيغة شبيهه بالتالي :

كود SQL

```sql
SELECT * Form Table1 WHERE Date > #12/03/04#;
```

والآن : ماذا لو أردنا البحث بتحقيق مجموعة شروط أو أحدها أو تحقيق شرط مع انتفاء آخر ؟

من أجل هذا الغرض نستخدم المعاملات المنطقية البسيطة And و Or كما تعلمنا في لغات البرمجة.

والآن إلى مثال سريع ، سنبحث عن الأشخاص الذين يكبر عمرهم عن 17 بشرط ألا يكونوا متزوجين :

كود SQL

```sql
SELECT Fname FROM tb_main WHERE age > 17 AND marry = false;
```

استخدام المعامل LIKE :

نستخدم المعامل LIKE للبحث عن الكلمات المشابهه لتعبير معين ...

ونستخدم للمعامل LIKE الصيغة التالية :

كود SQL

```sql
SELECT * FROM Table WHERE Field LIKE '%Name%';
```

نستخدم % للدلالة على وجود أحرف ما ... وقد نستخدمها في البداية ، أو النهاية ، أو كليهما .

وسيتضح الأمر في الأمثلة:

لعرض الأشخاص الذين قد تحتوي أسماءهم الأولى على hm ... سنستخدم جملة استعلام بهذا الشكل:

كود SQL

```sql
SELECT * FROM Tb_Main WHERE FName like '%hm%';
```

أما لو أردنا البحث عن الأشخاص الذين تبدأ أسماءهم بحرف A لذا لن نضع % قبل كلمة البحث :

كود SQL

```sql
SELECT * FROM Tb_Main WHERE FName like 'A%';
```

ولو أردنا البحث عن الأشخاص الذين ينتهي اسمهم بحرف معين سنضع % في البداية دون النهاية.

ملاحظة
في MS Access نستخدم * بدلاً من %

مما سبق نستنتج أن % تعني أي عدد من الحروف ، لكن ماذا لو أردنا تحديد عدد الحروف ؟
في هذه الحالو نستخدم "_" ، مناظرها في Access هو علامة الإستفهام "؟" .

الإستخدام	الرمز
أي عدد من الأحرف ابتداء من صفر فصاعداً	%
حرف واحد فقط .	_ (underscore)
حرف واحد فقط من مدى معين ، مثلاً حرف بين ال A وال N	[ ] Like '[A-N]ack'
حرف واحد فقط شريطة ألا يكون هو المحدد ، مثلاً حرف لا يكون B	[^] Like '[^B]ack'

الجدول 17. 3. بعض الأمثلة لاستعمال المعامل LIKE في عمليات المقارنة.

إذن : ماذا لو أردنا أن نبحث عن الأشخاص الذين تبدأ أسماءهم بحرف A بشرط أن يكون عدد حروفهم 3 فقط.

كود SQL
```sql
SELECT * FROM Tb_Main WHERE FName like 'A__';
```

استخدام المعامل BETWEEN

نستخدم هذا المعامل للبحث ضمن نطاق معين ، وأشهر استعمال لهذا المعامل هو استخدامه في البحث ضمن التواريخ.

يأخذ البحث باستخدام BETWEEN الصيغة التالية:

كود SQL
```sql
SELECT colum FROM table WHERE field BETWEEN v1 AND v2;
```

فمثلاً لاستخراج الأشخاص الذين تتروواح أعمارهم بين 20 و 60:

كود	SQL

```
SELECT *
FROM tb_main
WHERE age BETWEEN 20 AND 60;
```

ويمكن استخدام نفس الطريقة مع الأسماء، والتواريخ بطبيعة الحال.

الدوال في الاستعلامات.

`تستخدم العديد من الدوال ضمن طيات جمل الاستعلام ، وهي شائعة الاستخدام ، ومريحة ، وتعيد قيمة وحيدة - لا تعيد جدول - سنتعرف على بعض الدوال مع بعض الأمثلة خلال هذا الدرس.

ولننظر نظرة سريعة إلى الصيغة العامة لاستخدام الدوال والتي تأخذ الشكل التالي:

كود	SQL

```
SELECT func(colum) FROM table WHERE condition;
```

الدالة AVG:

تعطينا هذه الدالة متوسط حقل ما ، ولحساب متوسط الأعمار في قاعدة البيانات مثلاً نكتب أمراً كالتالي:

كود	SQL

```
SELECT AVG(Age) FROM tb_main;
```

ولك ان تتخيل ماذا سنفعل لو لم نستخدم هذه الدالة ، كنا سندور على جميع السجلات حيث نجمع ارقام كل سجل ثم ننتقل إلى التالي وهكذا - برمجياً. -

أيضاً يمكننا استخدام الدالة بشرط ... فمثلاً لحساب متوسط أعمار الأشخاص الذين لا تزيد أعمارهم عن 25:

كود	SQL

```
SELECT AVG(Age) FROM tb_main WHERE age < 25;
```

لو لاحظت لوجدت أن ناتج الدالة يظهر في حقل تحت اسم Expr100 أو ما شابه ... لذا قم - إذا كنت تريد - باعادة تسمية حقل الناتج عن طريق As كما تعلمنا سابقاً وبالشكل التالي:

كود SQL
```sql
SELECT AVG(Age) AS AVGAGE FROM tb_main WHERE age < 25;
```

ملاحظة

لاحظ أن الدالة سوف تتجاهل السجلات الفارغة...

الدوال Sum,Max,Min :

تعطي هذه الدوال المجموع - الأكبر - الأصغر على التوالي ، وهذا مثال على أكبر عمر في قاعدة البيانات:

كود SQL
```sql
SELECT max(Age) AS mxAGE FROM tb_main;
```

ملاحظة

لا تنس أن بإمكاننا وضع شرط لعملية الإستعلام.

الدالة Count:

وتعيد هذه الدالة عدد السجلات ، لنعيد مثلاً عدد السجلات :

كود SQL
```sql
SELECT COUNT(*)
FROM Table1;
```

ولتجاهل التكرارات :

كود SQL
```sql
SELECT COUNT(DISTINCT Title)
FROM Table1;
```

العلاقات :

كانت هذه بعض الدوال البسيطة والاكثر شهرة في TSql والآن ، ماذا لو كانت لدينا علاقات ونرغب في العمل عليها. في هذه الحالة سوف نستخدم الجملة بالشكل التالي :

كود	SQL

```
SELECT tb_main.Fname, tb_main.lname, Tb_R1.passport, tb_r1.bclass, tb_r1.from
FROM tb_main, tb_r1
WHERE tb_main.Number = tb_r1.Number;
```

دعنا الآن نلاحظ الفروق بين هذه الطريقة والطريقة السابقة:

أولاً : نقوم بكتابة اسم الجدول.اسم الحقل وذلك مهم لأننا نتعامل مع أكثر من جدول.

ثانياً : نقوم بكتابة اكثر من جدول بعد عبارة FROM لأننا نريد النواتج من أكثر من جدول.

ثالثاً : السطر الأخير من جملة الاستعلام هو لكي يعرض المعلومات التي تتشابه أرقامها في الجدولين سوية.

: DDL

ذكرنا أن DDL تختص بالتعامل مع هياكل قواعد البيانات ، سنبدأ أولاً بصيغة انشاء جدول وذلك بالشكل التالي:

كود	SQL

```
CREATE TABLE tb1 ( colum type, colum type,........);
```

وكمثال على ذلك ... لننشئ جدولاً تحت اسم Tb2 يحتوي على حقلين ، الإسم Name من نوع (نص - String) والآخر Age من نوع Number :

كود	SQL

```
CREATE TABLE tb1 (name text,age number) ;
```

إذن ماذا لو أردنا أن نحدد حجم حقل العمر Name بخمس خانات فقط ... إذن ضع عدد الخانات بين قوسين مع تعيين نوع بيانات الاسم ك Text وذلك بالشكل التالي:

كود	SQL

```
CREATE TABLE tb1 (name text(6),age number );
```

آخر نقطة سأشرحها في انشاء الجداول هي كيف نجعل الحقل لا يقبل فراغ Null - ...
ولنجرب هذا المثال مع الإسم أيضاً:

كود SQL
```
CREATE TABLE tb1 (name text(6) Not Null,age number );
```

ولكن ماذا عن الحذف ... في الواقع هذا الكود يقوم بحذف الجدول الذي قمنا بإنشاءه تواً:

كود SQL
```
DROP Table tb1;
```

ولو كان هذا الجدول مرتبطاً بعلاقات فلا بد من تحديثها باضافة CASCADE إلى آخر الكود وذلك بالشكل التالي:

كود SQL
```
DROP Table tb1 CASCADE;
```

والآن سنبدأ في اضافة حقول إليه وذلك بعد عملية الانشاء - أعتقد أن الفرق بين الجدول وقاعدة البيانات والحقول أصبح واضحاً. -
نستطيع اضافة الحقول بصيغة عامة بالطريقة التالية:

كود SQL
```
ALTER TABLE tb1 ADD colum type;
```

فلنضف حقلين جديدين : الأول هو تاريخ الميلاد والثاني النوع (ذكر - أنثى) :

كود SQL
```
ALTER TABLE tb1 ADD birth_Date date,Gender Text;
```

والآن لنحذف أحد هذه الحقول وليكن حقل ... Gender لاحظ أننا دائماً نضيف كلمة CASCADE لتحديث العلاقات - في حال وجودها - ، وذلك بالشكل التالي:

كود SQL
```
ALTER TABLE tb1 DROP gender CASCADE;
```

DML مرة أخرى:

بعد ان تعلمنا جزء البحث والاستعلام في DML سوف نتعلم الآن اضافة السجلات ، وسوف نستخدم Tb_Main كجدول لنجرب أوامرنا عليه ، والآن إلى الصيغة العامة لأمر الإضافة:

كود	SQL

```
INSERT INTO table VALUES (v1, v2,......);
```

لاحظ أننا سنمرر القيم واحداً وراء الثاني وتفصل بينهم فاصلة ، مع مراعاة وضع النصوص بين علامتي تنصيص "" والتواريخ بين علامتي##.

سنمرر القيم التالية : الرقم - الإسم الأول - الإسم الأخير - رقم الهاتف - العمر - العنوان - البريد الإلكتروني - السنة - اللغة - الموقع - متزوج/لا - ملاحظات.

وذلك بالشكل التالي:

كود	SQL

```
INSERT INTO tb_main VALUES
(11,'ahmed','gamal',6666666,18,'Cairo','Hammada2091','First
Year','','',false,'');
```

التعديل:

الصيغة العامة لتعديل بيانات سجل بالشكل التالي:

كود	SQL

```
UPDATE table_name
SET colum1 = v1 , colum2 = v2
WHERE colum = v;
```

سنجرب الآن مثالاً لجعل العمر = 28 والموقع = www.vb4arab.com للشخص الذي اسمه الأول(Ahmed) والاخير (Gamal) :

كود	SQL

```
UPDATE tb_main
SET age=25 , site='www.vb4arab.com'
WHERE fname='ahmed' AND lname='Gamal';
```

ولكن ماذا عن الحذف ... في الواقع تتخذ جملة البحث صيغة عامة كالتالي:

SQL كود

```sql
DELETE FROM table WHERE colum = value;
```

ويمكن أن يكون الحذف تحت تأثير تحقق أكثر من شرط .. في المثال التالي سوف نحذف جميع الأشخاص الذين تزيد أعمارهم عن الأربعين:

SQL كود

```sql
DELETE FROM tb_main WHERE age>40;
```

ولكن ماذا عن حذف كامل محتويات جدول ما ... هذه هي الطريقة:

SQL كود

```sql
DELETE * FROM table;
```

4. 3. Stored Procedures

ال Stored Prcoedure هو جملة استعلام مخزنة في قاعدة البيانات ، الموضوع بسيط جداً ، ولذا لنستعرض سوية جملة الإستعلام تلك :

SQL كود

```sql
SELECT [First Name] FROM employee_info WHERE age>21;
```

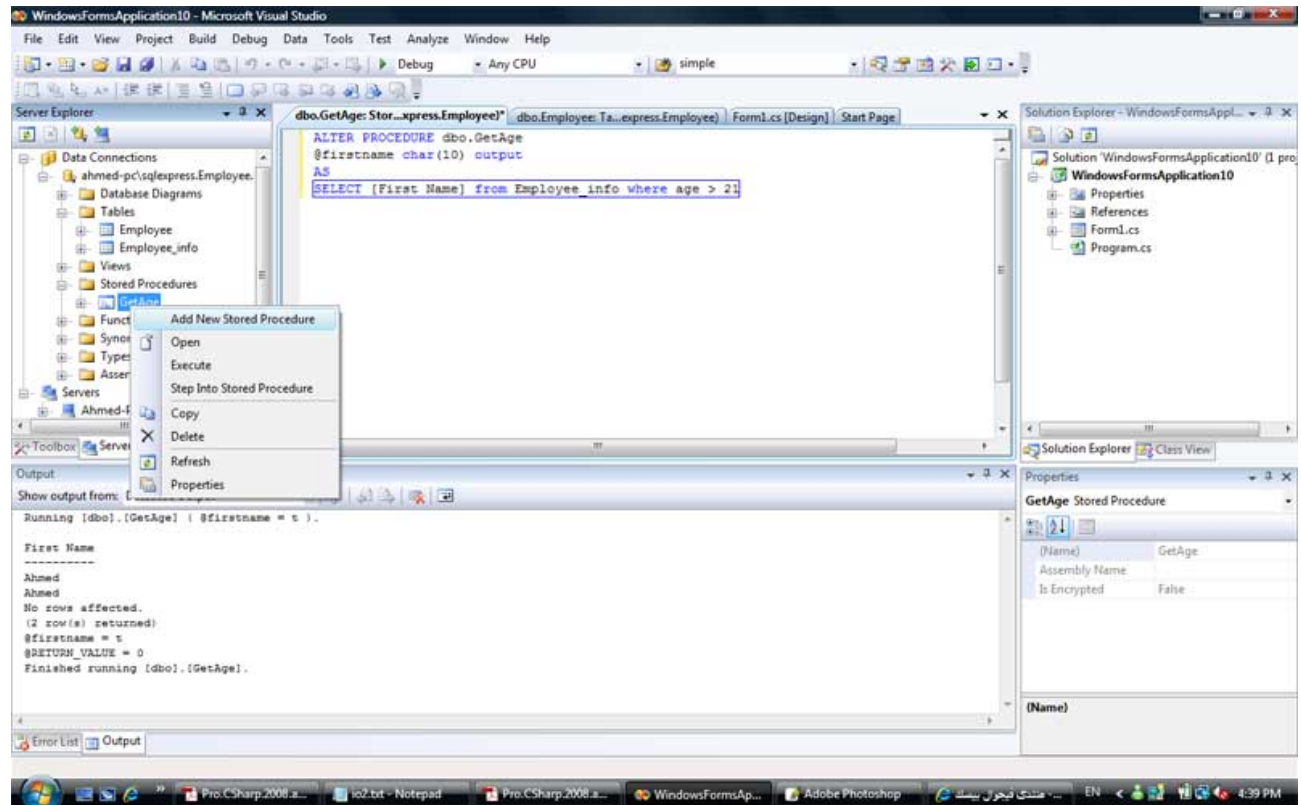
وهو ما يتم تحويله ل Procedure بالشكل التالي :

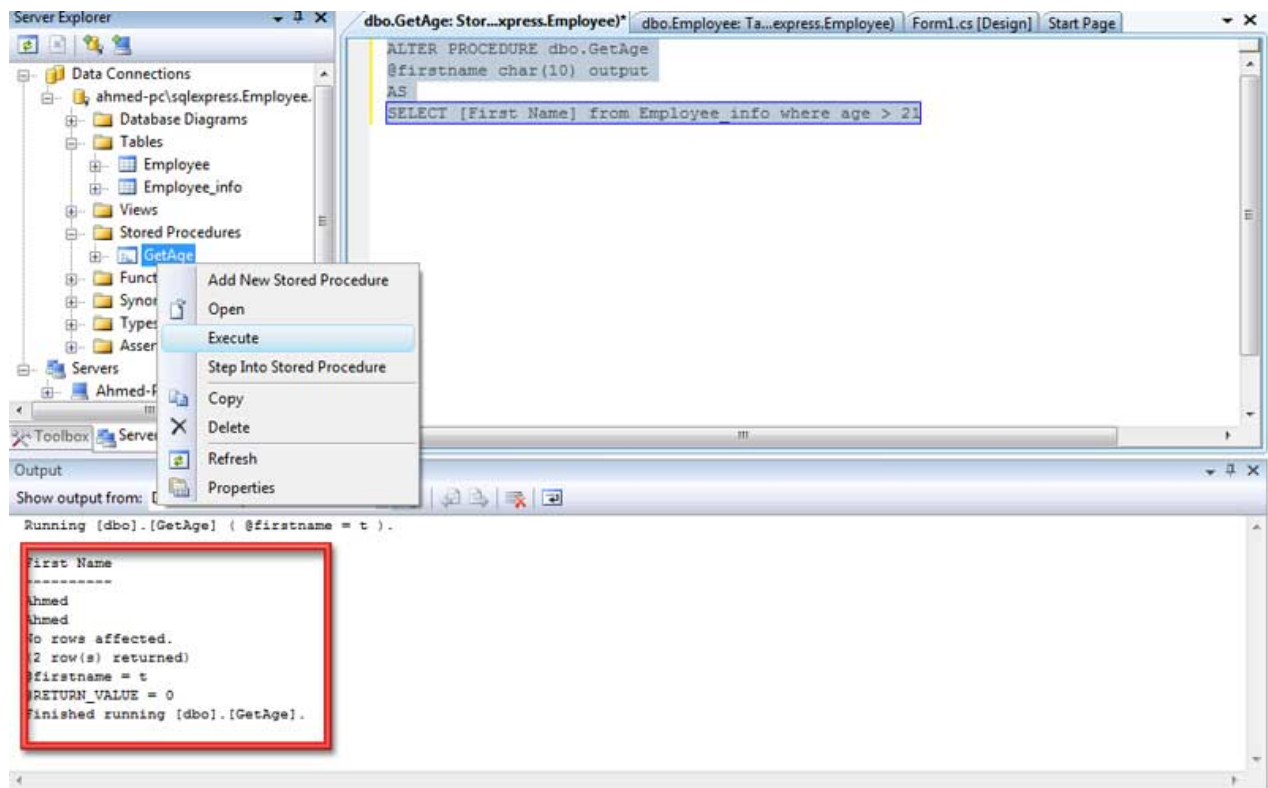
T-SQL كود

```sql
CREATE PROCEDURE dbo.GetAge
@firstname char(10) output

AS
SELECT @firstname =[First Name]
FROM Employee_info
WHERE age > 21
```

- السطر الأول Create أو Alter ثم اسم ال Stored Prcoedure .
- السطر الثاني يحتوي على البارميترات أو ال return value يتم تحديد نوعه وطريقته Input أو Output .

- السطر الثالث جملة الإستعلام .

والآن سنذهب إلى قسم Stored Procedure وسنختار انشاء واحد جديد ونكتب فيه هذه الاستعلام:

الصورة 17. 8. اضافة Stored Procedure.

يمكنك تجربة الناتج عن طريق اختيار Excute ، وسيكون الناتج شيئاً مثل هذا

الصورة 17. 9. نتائج تنفيذ ال Stored Procedure.

يمكنك جعل جملة الاستعلام تستقبل بارميتر لتحديد مثلاً الشرط المطلوب ، الشكل التالي كمثال:

كود	T-SQL

```
ALTER PROCEDURE dbo.GetAge
@condition int,
@firstname char(10) output
AS
SELECT @firstname =[First Name]
FROM Employee_info
WHERE age > @condition
```

في هذه الحالة يمكنك تمرير بارميتر سواء من البرنامج او حتى في الSQL Server لتقوم بتنفيذ جملة الاستعلام على اساسه .

طبعاً لا تنس أن بامكانك عمل نفس جملة الاستعلام Update او Delete أو Insert بدلاً من Select ، وحسب جملة الاستعلام المطلوبة.

4 .4. SQL Injection

لو كنت تريد معرفة لماذا نستخدم Stored Procedure بدلاً من جمل الاستعلام مباشرة فهذا هو الجزء الذي عليك أن تهتم به كثيراً .

لنفترض أنك داخل برنامجك تقوم بعملية تسجيل الدخول ، عن طريق اسم المستخدم وكلمة المرور :

كود C#

```
string cmd = "select id from Users where nams like '"+textBox1.Text+"' and pass like '"+textBox2.Text+"'";
```

الطبيعي أنه لو كان الإسم موجود فسوف تعيد الدالة قيمة ID أما في حالة العكس فلن تعود بنتيجة ، والآن لنفترض أن شخصاً ما قام بكتابة الجزء التالي في مربع النص الأول :

```
insert into Users values(1,'NewUser','1234–('
```

لن يدخل هذه المرة ، ولكنه سيقوم باضافة هذا المستخدم للجدول ، في المرة اللاحقة سيكون بامكانه الدخول باسم NewUser وبرقم سري 1234 !!!

أو ابسط من ذلك ، لو قام بكتابة الجملة التالية في مربعيي النص ، فهذا سيحقق الشرط :

```
a' or 't'='t
```

هل تخيلت حجم الضرر الذي يمكن أن يسببه هذا الموضوع ؟

يمكن أيضاً أن يقوم بكتابة:

```
drop table Tablename
```

وفقط !!!

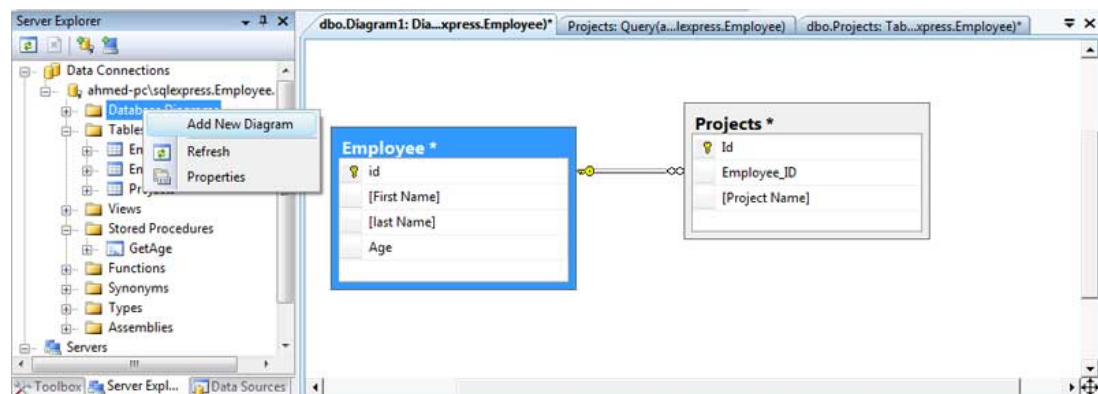
تفيدك الـ Stored Procedure في أن أي قيمة يتم ادخالها داخل المتغير تعامل معاملة متغير فقط ولا يمكنها التنفيذ أبداً ، ويمكننا عملها بطريقة أخرى سنتعرف عليها لاحقاً ...

4 .5. العلاقات

قم بعمل جدول فرعي نضع فيه مثلاً المشاريع التي يعمل عليها كل موظف، سيكون الناتج شيئاً مثل هذا:

	Id	Employee_ID	Project Name
	1	1	Traffic System
	2	2	GIS
	3	1	GIS
	4	4	Rowad
	5	5	Rowad
	6	6	Rowad
▸*	NULL	NULL	NULL

الصورة 17. 9. اضافة جدول جديد.

لربط العلاقات ، من القائمة الجانبية ستجد Database Diagram ، قم باختيار Add New وقم بسحب القيم التي تود ربطها في العلاقة بالشكل التالي مثلاً:

الصورة 17. 10. مخطط العلاقات بين الجداول في قاعدة البيانات Database Diagram.

هنا نكون قد انتهينا مما نحتاج إليه كثيراً في Sql Server لو كنت تحب التعرف على المزيد في عالم SQL Server يمكنك البدء من هنا:

رابط

http://msdn.microsoft.com/en-us/sqlserver/default.aspx

والآن سنعود مرة أخرى للغة البرمجة ...

5. الوضع المتصل

5. 1. التعامل مع ال ConnectionStringBuilder

يوفر لك هذا الكائن طريقة تفصيلية لبناء ال Connection String الخاص بك ، هذا المثال يوضح الأكثر استخداماً:

كود C#

```
SqlConnectionStringBuilder cnStrBuilder = new SqlConnectionStringBuilder();
cnStrBuilder.InitialCatalog = "Employee";
cnStrBuilder.DataSource = @"(local)\SQLEXPRESS";
cnStrBuilder.ConnectTimeout = 30;
cnStrBuilder.Password = "124";
cnStrBuilder.UserID = "Ahmed";
SqlConnection cn = new SqlConnection();
cn.ConnectionString = cnStrBuilder.ConnectionString;
cn.Open();
```

كود VB.NET

```
Dim cnStrBuilder As New SqlConnectionStringBuilder()
cnStrBuilder.InitialCatalog = "Employee"
cnStrBuilder.DataSource = "(local)\SQLEXPRESS"
cnStrBuilder.ConnectTimeout = 30
cnStrBuilder.Password = "124"
cnStrBuilder.UserID = "Ahmed"
Dim cn As New SqlConnection()
cn.ConnectionString = cnStrBuilder.ConnectionString
cn.Open()
```

حيث تجد:

InitialCatalog لتحديد اسم قاعدة البيانات

DataSource لتحديد مسار قاعدة البيانات

ConnectTimeout لتحديد الوقت الذي يمكن استغراقه من اجل محاولة الوصول إلى قاعدة البيانات او ايقاف العملية عند انتهاءه.

- Password كلمة مرور قاعدة البيانات إن وجدت.

- UserID اسم المستخدم لقاعدة البيانات إن وجد.

هناك العديد من العناصر الأخرى أيضاً يمكنك استعراضها من هنا:

رابط

http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlconnectionstringbuilder_members.aspx

2. 5. التعامل مع الفئة Command

ال Command هي جملة الاستعلام التي نستخدمها لتنفيذ اي نوع من العمليات على قاعدة البيانات، تنقسم إلى ثلاث انواع رئيسية:

- StoredProcedure
- TableDirect
- Text

في المثال السابق استخدمنا ال Command كجملة استعلام نصية مباشرة Text، كانت بالشكل التالي مثلاً:

كود C#

```
string strSQL = "Select * From Employee_Info";
SqlCommand myCommand = new SqlCommand(strSQL, cn);
```

كود VB.NET

```
Dim strSQL As String = "Select * From Employee_Info"
Dim myCommand As New SqlCommand(strSQL, cn)
```

أو بهذا الشكل:

كود C#

```
SqlCommand testCommand = new SqlCommand();
testCommand.Connection = cn;
testCommand.CommandText = strSQL;
```

كود	VB.NET

```
Dim testCommand As New SqlCommand()
testCommand.Connection = cn
testCommand.CommandText = strSQL
```

اثناء التنفيذ قمنا بربطها مباشرة باستخدام ExecuteReader ، في الواقع هناك عدة طرق عدة طرق للتنفيذ:

ExecuteReader: في حالة كون الناتج عدد كبير من البيانات ، يتم تعريف DataReader وربط الناتج به لقراءته ، وهو ما تعرفنا عليه في مراحل سابقة.

ExecuteNonQuery: في حالة عدم وجود نواتج اصلاً ، مثل تعريف عملية Update أو Delete حيث الناتج الوحيد هو تنفيذ العملية من عدمه ، سيتم التعرف عليه لاحقاً.

ExecuteScalar: في حالة كون الناتج وحيد ، مثل الاستعلام عن فقط عن الاسم الأول للشخص صاحب الرقم القومي xxxxxx.

ExecuteXmlReader : تنفيذ الناتج واعادته على شكل XML يتم تعريف XmlReader وربط الناتج به لقراءته ، يمكنك الرجوع إلى دروس XML لمعرفة المزيد عن XmlReader.

استخدام ال Parameterized Command Objects

كما لاحظت من الدرس الذي تحدثنا فيه عن ال SQL Injection ، فإن الطريقة التقليدية لجمل الاستعلام تظل خطرة ، لذا نبدا باستخدام Parameters لجمل الاستعلام لدينا ايضاً حتى لو لم نكن نتعامل مع Stored Procedure.

لو أخذنا المثال التالي للتجربة:

كود	C#

```
string strSQL = "Select [First Name]+[Last Name] as [Full Name], Age From
Employee where ID=" + ID;
SqlCommand myCommand = new SqlCommand(strSQL, cn);
```

كود	VB.NET

```
Dim strSQL As String = "Select [First Name]+[Last Name] as [Full Name], Age
From Employee where ID=" + ID
Dim myCommand As New SqlCommand(strSQL, cn)
```

ستجد اننا قادرين بمبادئ ال SQL Injection من اختراق هذا النظام بسهولة ، لذا البديل يكون باستخدام وتعريف SqlParameter:

كود C#
```
string sql = string.Format("Select [First Name]+[Last Name] as [Full Name], Age
From Employee where ID=@ID");
using (SqlCommand cmd = new SqlCommand(sql, this.sqlCn))
{
    // Fill params collection.
    SqlParameter param = new SqlParameter();
    param.ParameterName = "@ID";
    param.Value = 10;
    param.SqlDbType = SqlDbType.Int;
    cmd.Parameters.Add(param);
    // هنا شكل بأي الاستعلام جملة تنفيذ
    cmd.ExecuteNonQuery();
}
```

كود VB.NET
```
Dim sql As String = String.Format("Select [First Name]+[Last Name] as [Full
Name], Age From Employee where ID=@ID")
        Using cmd As New SqlCommand(sql, Me.sqlCn)
' Fill params collection.
Dim param As New SqlParameter()
            param.ParameterName = "@ID"
            param.Value = 10
            param.SqlDbType = SqlDbType.Int
            cmd.Parameters.Add(param)

' هنا شكل بأي الاستعلام جملة تنفيذ
            cmd.ExecuteNonQuery()
End Using
```

هنا كما لاحظت قمنا بتمرير باميترات إلى جملة استعلام قمنا نحن بكتابتها ضمن البرنامج.

الحالة الثانية التي لدينا وهي الشائعة الاستخدام هي حالة تعريف Stored Procedure

لو افترضنا مثلاً جملة الاستعلام التي أنشأناها في أول درس لنا بالشكل التالي:

كود	T-SQL

```
ALTER PROCEDURE dbo.GetAge
@condition int,
@firstname char(10) output
AS
SELECT @firstname=[First Name] from Employee_info where age > @condition
WHERE age > @condition
```

وقمنا بحفظها باسم GetAge ، الآن نريد استدعاءها من البرنامج ، يتم ذلك بالشكل التالي مثلاً

كود	C#

```
using (SqlCommand cmd = new SqlCommand("GetAge", cn))
{
    cmd.CommandType = CommandType.StoredProcedure;
    SqlParameter param = new SqlParameter();
    param.ParameterName = "@condition";
    param.SqlDbType = SqlDbType.Int;
    param.Value = myAge;
    param.Direction = ParameterDirection.Input;
    cmd.Parameters.Add(param);

    param = new SqlParameter();
    param.ParameterName = "@firstnameName";
    param.SqlDbType = SqlDbType.Char;
    param.Size = 10;
    param.Direction = ParameterDirection.Output;
    cmd.Parameters.Add(param);

    cmd.ExecuteNonQuery();
    MessageBox.Show(cmd.Parameters["@firstName"].Value.ToString());
}
```

كود	VB.NET

```
Using cmd As New SqlCommand("GetAge", cn)
    cmd.CommandType = CommandType.StoredProcedure

    Dim param As New SqlParameter()
    param.ParameterName = "@condition"
    param.SqlDbType = SqlDbType.Int
    param.Value = myAge
    param.Direction = ParameterDirection.Input
    cmd.Parameters.Add(param)

    param = New SqlParameter()
    param.ParameterName = "@firstnameName"
    param.SqlDbType = SqlDbType.[Char]
    param.Size = 10
    param.Direction = ParameterDirection.Output
    cmd.Parameters.Add(param)

    cmd.ExecuteNonQuery()
    MessageBox.Show(cmd.Parameters("@firstName").Value.ToString())
End Using
```

كما لاحظت ، قمنا بتعريف نوع ال Command هنا نظراً لإن الافتراضي هو Text، ومن ثم قمنا بتعريف متغير الدخول ومتغير الخروج أيضاً.

5. 3. التعامل مع DataReaders

كما شاهدت في المثال السابق ، يمكن قراءة البيانات من DataReader بالشكل التالي مثلاً:

كود	C#

```
string listItem = "";
while (myDataReader.Read())
{
    listItem = "Full Name: " + myDataReader["Full Name"].ToString() + " Age: "
+ myDataReader["Age"].ToString();
    listBox1.Items.Add(listItem);
}
```

كود	VB.NET

```
Dim listItem As String = ""
While myDataReader.Read()
    listItem = "Full Name: " + myDataReader("Full Name").ToString() + " Age: "
+ myDataReader("Age").ToString()
    listBox1.Items.Add(listItem)
End While
```

حيث تعود الدالة Read ب true ما دام هناك سجلات للقراءة، وفي نفس الوقت مع كل استدعاء لها تنتقل إلى السجل التالي...

طريقة القراءة تكون بتحديد الحقل المراد قراءته myDataReader["Age"] او بتحديد رقمه في الترتيب myDataReader[2] مثلاً.

الخاصية FieldCount تعطينا عدد النتائج المعادة، لذا يمكننا تنفيذ نفس العملية السابقة بالشكل التالي:

كود | C#

```
for (int i = 0; i < myDataReader.FieldCount; i++)
{
    listItem = "Full Name: " + myDataReader["Full Name"].ToString() + " Age: "
+ myDataReader["Age"].ToString();
    listBox1.Items.Add(listItem);
    myDataReader.Read();
}
```

كود | VB.NET

```
For i As Integer = 0 To myDataReader.FieldCount - 1
    listItem = "Full Name: " + myDataReader("Full Name").ToString() + " Age: "
+ myDataReader("Age").ToString()
    listBox1.Items.Add(listItem)
   myDataReader.Read();
Next
```

NextResult :

تمكنك ال DataReader من تعريف جملتي استعلام لاعادة الناتج ، فمثلاً لاعادة اسماء الموظفين ثم اسماء المشاريع:

كود | C#

```
string strSQL = "Select * From Employee_info;Select * from projects";
SqlCommand myCommand = new SqlCommand(strSQL, cn);
do
{
    while (myDataReader.Read())
    {
        for (int i = 0; i < myDataReader.FieldCount; i++)
            listItem = "Data: " + myDataReader[0].ToString();
    }
}
while (myDataReader.NextResult());
```

كود	VB.NET

```
Dim strSQL As String = "Select * From Employee_info;Select * from projects"
Dim myCommand As New SqlCommand(strSQL, cn)
Do
    While myDataReader.Read()
        For i As Integer = 0 To myDataReader.FieldCount - 1
            listItem = "Data: " + myDataReader(0).ToString()
        Next
    End While
Loop While myDataReader.NextResult()
```

6. Data Access Layer

في التطبيقات الجدية ، لا يتم وضع الكود مع المظهر مع سيناريو وعمليات البرنامج اضافة لطبقة البيانات، بل يتم فصل كل منها في طبقة منفصلة وهو ما يعرف باسم Layers ، لمعرفة المزيد عن هذا الموضوع يمكنك مراجعة الرابط التالي:

رابط

http://www.al-asiri.com/ShowRecord.aspx?Action=Open&id=cefa426c-d9e0-4625-a66b-87fd6082ff89

وهناك تطبيق ايضاً هنا:

رابط

http://vb4arab.com/vb/showthread.php?t=10969

في هذه المرحلة ، سنحاول عمل data layer تكون خاصة بالتعامل مع قواعد البيانات لتحقيق نقطتين مهمتين:

- اعادة استخدامها اكثر من مرة.

- في حالة وجود خطأ يتم تحديد مصدره بسهولة ، وفي حالة التعديل لا يتم التعديل سوى على هذه الطبقة.

وفي هذه الحالة لن يتطرق مبرمج اي جزء من البرنامج إلى كيفية الاتصال بقاعدة البيانات ، حيث ان كل ما لديه هي مجموعة من الدوال للاتصال والحذف والتعديل وكل العمليات التي يريدها مع بارميترات مناسبة دون التدخل في كيفية الاتصال بقواعد البيانات تماماً.

يمكنك عمل هذه الطبقة بعدة طرق ، ابسطها استخدام بعض البرامج الجاهزة التي تقوم بهذه العملية وتقوم بعمل استخراج لهذه ال layer حسب لغة البرمجة التي تحددها ، ايضاً يوفر لك الدوت نت طريقة لهذه العملية ، الطريقة الثانية هي الطريقة اليدوية...

في العجالة التالية سنتعرف على مثال بسيط لهذه العملية من اجل الموظفين ، يمكن تطبيق نفس المفاهيم على البرامج الجدية لاحقا.

1- عمليات فتح واغلاق قواعد البيانات

كود	C#

```
private SqlConnection cn = new SqlConnection();
public void OpenConnection(string connectionString)
{
    cn.ConnectionString = connectionString;
    cn.Open();
}
public void CloseConnection()
{
    cn.Close();
}
```

كود	VB.NET

```
Private cn As New SqlConnection()
Public Sub OpenConnection(ByVal connectionString As String)
    cn.ConnectionString = connectionString
    cn.Open()
End Sub
Public Sub CloseConnection()
    cn.Close()
End Sub
```

2- سيناريو عملية الحذف.

في هذه العملية سنتيح للمستخدم حذف الموظف باسمه ، او حذف الموظف برقمه ، او حذف الموظف مثلاً بدلالة العمر وبعدة اختيارات سواء اكبر او اقل او يساوي مثلاً مع استخدام مبدأ الOverLoading ومبدأ عمل Enums ايضاً واللذان تم شرحهما في دروس سابقة.

*** لاحظ ان مهمتك في هذه المرحلة هي عمل كل الدوال التي يمكن استخدامها في البرنامج تحت اي ظرف من الظروف

C# كود

```
public void DeleteEmployee(int id){
    string sql = string.Format("Delete from Employee where ID = {0}",id);
    using (SqlCommand cmd = new SqlCommand(sql, this.sqlCn))
    {
        try{
            cmd.ExecuteNonQuery();
        }
        catch (SqlException ex) {
            Exception error = new Exception("some error occures: ", ex);
            throw error;
        }
    }
}
public void DeleteEmployee(string name){
    string sql = string.Format("Delete from Employee where [First Name] =
'{0}'", name);
    using (SqlCommand cmd = new SqlCommand(sql, this.sqlCn))    {
        try{
            cmd.ExecuteNonQuery();
        }
        catch (SqlException ex){
            Exception error = new Exception("some error occures: ", ex);
            throw error;
        }
    }
}
enum deletecondition{
    morethan = 0,
    lessthan = 1,
    equal = 2
}
public void DeleteEmployee(int age, deletecondition delcondition){
    string sql = "";
    if (delcondition == deletecondition.morethan)
        sql = string.Format("Delete from Employee where age > {0}", age);
    else if (delcondition == deletecondition.lessthan)
        sql = string.Format("Delete from Employee where age < {0}", age);
    else
        sql = string.Format("Delete from Employee where age = {0}", age);
    using (SqlCommand cmd = new SqlCommand(sql, this.sqlCn))
    {
        try
        {
            cmd.ExecuteNonQuery();
        }
        catch (SqlException ex)
        {
            Exception error = new Exception("some error occures: ", ex);
            throw error;
        }
    }
}
```

VB.NET كود

```
Public Sub DeleteEmployee(ByVal id As Integer)
    Dim sql As String = String.Format("Delete from Employee where ID = '{0}'",
id)
    Using cmd As New SqlCommand(sql, Me.sqlCn)
        Try
            cmd.ExecuteNonQuery()
        Catch ex As SqlException
            Dim [error] As New Exception("some error occures: ", ex)
            Throw [error]
        End Try
    End Using
End Sub
Public Sub DeleteEmployee(ByVal name As String)
    Dim sql As String = String.Format("Delete from Employee where [First Name]
= '{0}'", name)
    Using cmd As New SqlCommand(sql, Me.sqlCn)
        Try
            cmd.ExecuteNonQuery()
        Catch ex As SqlException
            Dim [error] As New Exception("some error occures: ", ex)
            Throw [error]
        End Try
    End Using
End Sub
Enum deletecondition
    morethan = 0
    lessthan = 1
    equal = 2
end enum

Public Sub DeleteEmployee(ByVal age As Integer, ByVal delcondition As
deletecondition)
    Dim sql As String = ""
    If delcondition = deletecondition.morethan Then
        sql = String.Format("Delete from Employee where age > {0}", age)
    ElseIf delcondition = deletecondition.lessthan Then
        sql = String.Format("Delete from Employee where age < {0}", age)
    Else
        sql = String.Format("Delete from Employee where age = {0}", age)
    End If
    Using cmd As New SqlCommand(sql, Me.sqlCn)
        Try
            cmd.ExecuteNonQuery()
        Catch ex As SqlException
            Dim [error] As New Exception("some error occures: ", ex)
            Throw [error]
        End Try
    End Using
End Sub
```

3- سيناريو عمليات الاضافة والتعديل

بنفس الطريقة يمكن التعديل بعدة خيارات او الاضافة بعدة طرق.

بالنسبة لعملية الاضافة سنجبره على ادخال الاسم الأول والأخير فقط.

والتعديل ينبغي أيضاً أن يكون بنفس الصورة ، ولكن منعاً للاطالة سنعدل بدلالة الرقم الاسم الأول فقط.

كود	C#

```
public void InsertEmployee(string fname, string lname, int age)
{
    // Format and execute SQL statement.
    string sql = string.Format("Insert Into Employee_info" +
    "([First Name], [Last Name]) Values" +
    "('{0}', '{1}')", fname, lname);
    using (SqlCommand cmd = new SqlCommand(sql, this.sqlCn))
    {
        cmd.ExecuteNonQuery();
    }
}
public void UpdateEmployee(int id, string newFirstName) {
    string sql = string.Format("Update Employee Set [First Name] = '{0}' Where
ID = '{1}'",
    newFirstName, id);
    using (SqlCommand cmd = new SqlCommand(sql, this.sqlCn)) {
        cmd.ExecuteNonQuery();
    }
}
```

كود	VB.NET

```
Public Sub InsertEmployee(ByVal fname As String, ByVal lname As String, ByVal
age As Integer)
    ' Format and execute SQL statement.
    Dim sql As String = String.Format("Insert Into Employee_info" + "([First
Name], [Last Name]) Values" + "('{0}', '{1}')", fname, lname)

    Using cmd As New SqlCommand(sql, Me.sqlCn)
        cmd.ExecuteNonQuery()
    End Using
End Sub
Public Sub UpdateEmployee(ByVal id As Integer, ByVal newFirstName As String)
    Dim sql As String = String.Format("Update Employee Set [First Name] = '{0}'
Where ID = '{1}'", newFirstName, id)
    Using cmd As New SqlCommand(sql, Me.sqlCn)
        cmd.ExecuteNonQuery()
    End Using
End Sub
```

4- سيناريو عمليات البحث.

قبل البدء في سيناريو عملية البحث ، نود أن نشير إن الدوال السابقة ينقصها شيء هام وهي عملية ال Parameters لتلافي المشاكل الناتجة عن الSql Injection، لكن كانت الامثلة السابقة للتوضيح فقط ، في عملية البحث الآن سنطبق ما تعلمناه لحل هذه المشكلة.

سنجرب عملية بحث واحدة عن الاسم الأول والأخير للأشخاص برقم ID معين ، لا تنسى أنك مطالب في Data Layer بعمل كل الطلبات التي قد يحتاجها مبرمج عمليات البرنامج لكي لا يحتاج لكتابة حتى جملة استعلام واحدة.

سنقوم اولاً بعمل Stored Procedure :

كود	T-SQL

```
CREATE PROCEDURE GetFirstNameByID
@id int,
@fName char(10) output
AS
SELECT @fName=[First Name] FROM Employee_info WHERE ID > @id
```

ومن ثم نقوم نعرف الدالة الخاصة بعملية البحث بالشكل التالي:

كود	C#

```
public string SelectName(int id)
{
    using (SqlCommand cmd = new SqlCommand("GetFirstNameByID", cn))
    {
        cmd.CommandType = CommandType.StoredProcedure;
        SqlParameter param = new SqlParameter();
        param.ParameterName = "@id";
        param.SqlDbType = SqlDbType.Int;
        param.Value = ID;
        param.Direction = ParameterDirection.Input;
        cmd.Parameters.Add(param);
        param = new SqlParameter();
        param.ParameterName = "@fName";
        param.SqlDbType = SqlDbType.Char;
        param.Size = 10;
        param.Direction = ParameterDirection.Output;
        cmd.Parameters.Add(param);
        cmd.ExecuteNonQuery();
        return cmd.Parameters["@fName"].Value.ToString();
    }
    return carPetName;
}
```

VB.NET | كود

```
Public Sub InsertEmployee(ByVal fname As String, ByVal lname As String, ByVal
age As Integer)
    ' Format and execute SQL statement.
    Dim sql As String = String.Format("Insert Into Employee_info" + "([First
Name], [Last Name]) Values" + "('{0}', '{1}')", fname, lname)

    Using cmd As New SqlCommand(sql, Me.sqlCn)
        cmd.ExecuteNonQuery()
    End Using
End Sub

Public Sub UpdateEmployee(ByVal id As Integer, ByVal newFirstName As String)

    Dim sql As String = String.Format("Update Employee Set [First Name] = '{0}'
Where ID = '{1}'", newFirstName, id)
    Using cmd As New SqlCommand(sql, Me.sqlCn)
        cmd.ExecuteNonQuery()
    End Using
End Sub
```

-لاحقاً يمكنك فصل ال DataLayer حتى في dll منفصلة لضمان تشغيلها مع اكثر من تطبيق ، في الاستخدام لاحقاً وعلى افتراض ان الفئة Class لطبقة البيانات DataLayer تحمل الاسم myDataLayer، ولتطبيق عملية مثل الحذف كل المطلوب منك هو سطرين مثل الآتي:

C# | كود

```
myDataLayer example=new myDataLayer();
example.OpenConnection(myconnectionstring);
example.DeleteEmployee(10);
example.DeleteEmployee("Ahmed");
```

VB.NET | كود

```
Dim example As New myDataLayer()
example.OpenConnection(myconnectionstring)
example.DeleteEmployee(10)
example.DeleteEmployee("Ahmed")
```

وفقط!!!

وبنفس النظام لو كانت dll قم باستيرادها في برنامجك ثم قم باستخدامها بنفس الطريقة.

لو لاحظت ، بهذه الطريقة اصبح المبرمج لسيناريو البرنامج ولباقي عملياته بعيداً كل البعد عن قواعد البيانات ، كما ان تقسيم العمل أصبح أوضح وبالتالي اصبح بالامكان تدارك المشاكل بصورة أوضح.

7. Asynchronous Data Access

في هذه الجزئية سنحاول تعلم طريقة تمنعنا من عمل عدة عمليات على قواعد البيانات في نفس الوقت لمنع التضارب ، ومع ان المشكلة ستواجهنا بصورة اكبر في الوضع المنفصل Disconnected والذي هو موضوع درسنا القادم ، إلا اننا سنجرب الحل في هذه العجالة السريعة ، جرب المثال التالي:

كود — C#
```
SqlConnection cn = new SqlConnection();
cn.ConnectionString = @"Data Source=(local)\SQLEXPRESS;Integrated
Security=SSPI;" +
"Initial Catalog=AutoLot;Asynchronous Processing=true";
cn.Open();
```

كود — VB.NET
```
Dim cn As New SqlConnection()
cn.ConnectionString = "Data Source=(local)\SQLEXPRESS;Integrated
Security=SSPI;" + "Initial Catalog=AutoLot;Asynchronous Processing=true"
cn.Open()
```

أول خطوة قمنا بها هي تعريف الوضع Asynchronous Processing = true في ال ConnectioString بهذه الطريقة اصبحنا قادرين على الاستفادة من الدوال التالية:

BeginExecuteReader() و EndExecuteReader()

BeginExecuteNonQuery() و EndExecuteNonQuery()

BeginExecuteXmlReader() و EndExecuteXmlReader()

سنجرب الآن على BeginExecuteReader و EndExecuteReader، وسنبدأ بتأخير جملة الاستعلام الأولى لخمس ثوان مثلاً:

كود — C#
```
string strSQL = "WaitFor Delay '00:00:05';Select * From Employee_info";
SqlCommand myCommand = new SqlCommand(strSQL, cn);
```

كود — VB.NET
```
Dim strSQL As String = "WaitFor Delay '00:00:05';Select * From Employee_info"
Dim myCommand As New SqlCommand(strSQL, cn)
```

هنا سنقوم بتنفيذ عمليات أخرى على ثريد آخر:

كود C#

```
IAsyncResult itfAsynch;
itfAsynch = myCommand.BeginExecuteReader(CommandBehavior.CloseConnection);
```

كود VB.NET

```
Dim itfAsynch As IAsyncResult
itfAsynch = myCommand.BeginExecuteReader(CommandBehavior.CloseConnection)
```

وتنفيذ بعض العمليات حتى الانتهاء من تنفيذ الثريد

كود C#

```
while (!itfAsynch.IsCompleted)
{
    // تغيير قيمة مؤشر مثلاً .
    System.Threading.Thread.Sleep(1000);
}
```

كود VB.NET

```
While Not itfAsynch.IsCompleted
' تغيير قيمة مؤشر مثلاً .
    Thread.Sleep(1000)
End While
```

الآن بما أننا خرجنا من ال Loop السابقة فهذا يعني انتهاء التنفيذ الأول، الآن سنقوم بتنفيذ العملية التي نريدها.

كود C#

```
SqlDataReader myDataReader = myCommand.EndExecuteReader(itfAsynch);
while (myDataReader.Read())
{
    MessageBox.Show(myDataReader[1].ToString());
}
myDataReader.Close();
```

كود VB.NET

```
Dim myDataReader As SqlDataReader = myCommand.EndExecuteReader(itfAsynch)
While myDataReader.Read()
    MessageBox.Show(myDataReader(0).ToString())
End While
myDataReader.Close()
```

8. Transactions

تعريف هذه العملية باختصار شديد هو وجود مجموعة من العمليات لا بد أن تتم سوية أو تتوقف سوية ، المثال الأشهر لهذه العملية هي عمليات التحويل البنكية من عميل1 إلى عميل2.

لذا خطوات العمل في قواعد البيانات لا بد أن تكون بالشكل التالي:

- سحب 500 دولار من حساب عميل 1.

- ايداع 500 دولار في حساب عميل 2.

وهاتان العمليتان لابد ان تتم سوية ، بمعنى لو تمت العملية الأولى ولم تتم العملية الثانية لانقطاع الاتصال مثلاً فهذا غير مقبول ، لذا لا بد ان يتم اعتماد العمليتين أو حذف اي عملية منهم تتم دون الأخرى وهو ما يعرف باسم roll back.

إذا قمنا بعمل transaction سيتم تعريف هذه المجموعة من العمليات في نظام قاعدة البيانات DBMS على شكل وحدة واحدة بحيث يتم تنفيذها سوية او عدم تنفيذها سوية.

الفئة المسؤولة عن هذه العملية هي الفئة SqlTransaction الموجودة ضمن مجال الأسماء System.Data.SqlClient، هناك فئات أخرى يمكنها تطبيق هذه العملية ضمن مجال أسماء .net ايضاً مثل:

System.EnterpriseServices: تتيح لنا هذه الفئة الاتصال مع مكونات COM+ التي تقدم لنا الدعم في هذه العملية.

System.Transactions: تتيح لنا هذه الفئة بناء تطبيقات تدعم الtransactions.

WCF و WWF : تمكننا هاتان الفئتان من تطبيق مبادئ ال transaction ايضاً.

حتى في نظم قواعد البيانات يمكنك القيام بهذه المهمة عن طريق تعريف Stored Procedure يقوم بعمل transactions عن طريق TRANSACTION و ROLLBACK و COMMIT ، يمكنك البدء في هذا النوع من هنا:

رابط

http://msdn.microsoft.com/en-us/library/ms187844.aspx

في ADO.net لدينا الفئة DBTransaction التي تطبق الواجهة IDbTransaction والذي يحتوي على الدوال الرئيسية التالية:

كود — C#

```
public interface IDbTransaction : IDisposable
{
    IDbConnection Connection { get; }
    IsolationLevel IsolationLevel { get; }
    void Commit();
    void Rollback();
}
```

كود — VB.NET

```
Public Interface IDbTransaction
    Inherits IDisposable
    ReadOnly Property Connection() As IDbConnection
    ReadOnly Property IsolationLevel() As IsolationLevel
    Sub Commit()
    Sub Rollback()
End Interface
```

يضيف لنا ال SqlTransaction دالة جديدة هي Save والتي تتيح لنا حفظ نقطة يتم الرجوع إليها في حالة الفشل في اتمام العملية بدل الرجوع في العملية بالكامل - لو كان هناك اجزاء من العملية أو العملية على مراحل.

ابسط مثال على تطبيق هذه العملية ، هو افتراض وجود جدول يحتوي على (اسم العميل - المبلغ المودع) وعندما نقوم بعملية تحويل من حساب إلى آخر نقوم بعملية بالشكل التالي:

كود — C#

```
SqlCommand cmdGet = new SqlCommand("update customers set total=total-" +
totalmoney.ToString() + " where ID" + custID.ToString(), cn);
cmdSelect.ExecuteNonQuery();
SqlCommand cmdSet = new SqlCommand("update customers set total=total+" +
totalmoney.ToString() + " where ID" + SuppID.ToString(), cn);
cmdSelect.ExecuteNonQuery();
```

كود	VB.NET

```
Dim cmdGet As New SqlCommand("update customers set total=total-" +
totalmoney.ToString() + " where ID" + custID.ToString(), cn)
cmdSelect.ExecuteNonQuery()

Dim cmdSet As New SqlCommand("update customers set total=total+" +
totalmoney.ToString() + " where ID" + SuppID.ToString(), cn)
cmdSelect.ExecuteNonQuery()
```

في هذه العملية ولأي سبب كان قد يتوقف الجزء الثاني من العملية ، لذا سنحاول اخبار نظام قاعدة البيانات بأننا سننفذ كلا العمليتين في نفس الوقت ، ولو لم يتم تنفيذ واحدة منهما سيتم الغاء الأخرى مباشرة.

كود	C#

```
SqlCommand cmdGet = new SqlCommand("update customers set total=total-" +
totalmoney.ToString() + " where ID" + custID.ToString(), cn);
SqlCommand cmdSet = new SqlCommand("update customers set total=total+" +
totalmoney.ToString() + " where ID" + SuppID.ToString(), cn);
SqlTransaction sqltr = null;
try
{
    sqltr = sqlCn.BeginTransaction();
    cmdGet.Transaction = sqltr;
    cmdSet.Transaction = sqltr;
    cmdGet.ExecuteNonQuery();
    cmdSet.ExecuteNonQuery();
    if (throwEx)
    {
        throw new ApplicationException("all operation canceld, some errors
occures");
    }
    sqltr.Commit();
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message);
    tx.Rollback();
}
```

كود VB.NET

```
Dim cmdGet As New SqlCommand("update customers set total=total-" +
totalmoney.ToString() + " where ID" + custID.ToString(), cn)
Dim cmdSet As New SqlCommand("update customers set total=total+" +
totalmoney.ToString() + " where ID" + SuppID.ToString(), cn)
Dim sqltr As SqlTransaction = Nothing
Try
    sqltr = sqlCn.BeginTransaction()

    cmdGet.Transaction = sqltr
    cmdSet.Transaction = sqltr

    cmdGet.ExecuteNonQuery()
    cmdSet.ExecuteNonQuery()
    If throwEx Then
        Throw New ApplicationException("all operation canceld, some errors
occures")
    End If
    sqltr.Commit()
Catch ex As Exception
    MessageBox.Show(ex.Message)
    tx.Rollback()
End Try
```

كما لاحظت ، نقوم ببدء عملية ال transaction ، وما لم يحدث أي خطأ فلن ندخل في الشرط ولذا سيتم تنفيذ دالة Commit لتنفيذ العمليتين واعتمادهما ، أما في حالة حدوث اي خطأ فسنقوم برمي Exception throw والذي ينقلنا مباشرة إلى catch لنقوم هناك باستدعاء الدالة Rollback من اجل الغاء جميع التأثيرات التي حدثت.

لو كنت تود تجربة هذا المثال لمعرفة كيفية حدوثة ، جرب جعل قيمة throwEx=true وجرب ما يحدث.

لو جربت مثالك الآن ستجد ان قيمة مبلغ العميل 1 مثلاً لم تتأثر ، أما في حالة عدم وجود خطأ في الجملة الثانية فستجد ان الكمية المحددة من المبلغ قد تم نقلها من عميل 1 إلى عميل 2.

***طبعاً في تطبيقاتك الجدية لن يكون نظام الايداع والصرف بهذا الشكل ، بل في العادة سيكون هناك جدول او اكثر من جدول يحتوي على العمليات المجراه لكل عميل من ايداع وصرف وسحب وخلافه.

9. الوضع المنفصل

في دروسنا السابقة اقتصر حديثنا في تقنية ADO.net عن الوضع المتصل connected layer وهو الذي ينشأ رابطة مباشرة بينك وبين قاعدة البيانات بحيث تؤثر جميع تعديلاتك فيها لحظياً ، الجزء الثاني الذي تقدمه لنا هذه التقنية هي امكانية تطبيق الوضع المنفصل disconnected layer والذي يسمح لك بالتعامل مع قاعدة البيانات في الذاكرة عن طريق مجموعة من الفئات ومن ثم نقل كافة التعديلات مرة واحدة إلى قاعدة البيانات.

يتم ذلك عن طريق عمل DataAdapter يكون وسيط بين قاعدة البيانات وبين البرنامج ، ومن ثم انشاء DataSet يتم العمل عليها ثم ارسالها مرة أخرى إلى قاعدة البيانات.

تحتوي ال DataSet على أي عدد من DataTable تحتوي على كائنات DataRow وDataColumn.

بقي ان نذكر بأن هذا الوضع لا يلزم فيه اي نوع من الاتصال مع قاعدة بيانات ، فال DataSet هي نفسها قاعدة بيانات لذا سنواصل دروسنا في البداية دون اي اتصال بقاعدة بيانات فعلية ثم نعود إليها مرة أخرى في نهاية هذا القسم .

9. 1. ال DataSet

باختصار شديد ال DataSet هي عبارة عن صورة لنظام قواعد بيانات ولكن في الذاكرة - مؤقت - ، المحتويات الأساسية لهذه الفئة هي:

الفئات الأساسية :

DataTableCollection : الجداول في هذه القاعدة.

DataRelationCollection : العلاقات بين الجداول المختلفة.

PropertyCollection : لاضافة خصائص إلى ال DataSet.

الخصائص الأساسية:

DataSetName : الاسم.

RemotingFormat : تحديد الطريقة التي يتم بها عمل serialize للمحتويات ، مثل binary او XML او غيره.

الدوال الأساسية:

AcceptChanges() : تطبيق التغييرات التي تمت على ال DataSet منذ آخر مرة تم عمل AcceptChanges لها إلى قاعدة البيانات.

RejectChanges() : الغاء جميع التعديلات التي تمت على ال DataSet منذ آخر مرة تم عمل AcceptChanges لها.

Clear() : مسح كافة محتويات الDataSet

Clone() : نسخ الهيكل structure لقاعدة البيانات بما فيها الجداول والعلاقات

Copy() : نسخ الهيكل اضافة إلى جميع البيانات.

GetChanges() : معرفة كافة التغييرات التي حدثت لقاعدة البيانات منذ آخر مرة تم عمل AcceptChanges لها.

HasChanges() : قيمة Boolean تحديد فيما اذا كانت هناك تغييرات قد تمت منذ آخر مرة تم عمل AcceptChanges ام لا.

Merge(): دمج عدة DataSet.

ReadXml() و WriteXml() : الكتابة والقراءة على شكل XML من الDataSet

انشاء DataSet :

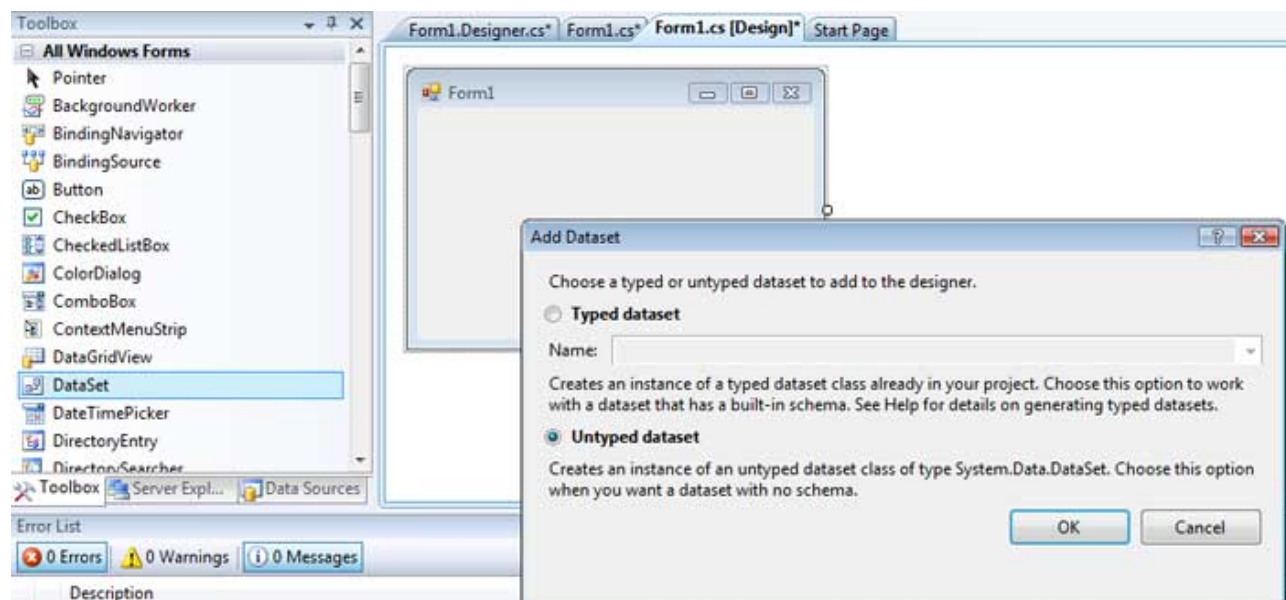
يمكنك انشاء DataSet عن طريق الكود بابسط طريقة بالشكل التالي:

كود	C#

```
DataSet empDataSet = new DataSet("My Employee");
```

كود	VB.NET

```
Dim empDataSet as DataSet = New DataSet("My Employee")
```

أو يمكنك انشاءها عن طريق أدوات net. ، من قائمة الأدوات Data ستجد DataSet قم بسحبها وسيظهر لك المعالج مباشرة بالشكل التالي:

الصورة 17. 11. اضافة DataSet من نافذة الأدوات.

9. 2. التعامل مع DataTable

الجدول هو العنصر الاساسي والذي سيحتوي لاحقاً على صفوف وأعمدة ، الخصائص والعناصر الأساسية له هي:

Copy() : نسخ هيكل الجدول.

DataSet : الDataSet الذي يحتوي على هذا الTable

PrimaryKey : ال PrimaryKey لهذا الجدول.

TableName : اسم الجدول.

ParentRelations : العلاقات التي يتحويها هذا الجدول .

9. 3. انشاء ال DataTable

كود C#
```
DataTable EmployeeTable = new DataTable();
EmployeeTable.PrimaryKey = new DataColumn[] { EmployeeTable.Columns[0] };
```

كود VB.NET
```
Dim EmployeeTable As New DataTable()
EmployeeTable.PrimaryKey = New DataColumn() {EmployeeTable.Columns(0)}
```

اضافة DataTable إلى DataSet :

كود C#
```
empDataSet.Tables.Add(EmployeeTable);
```

كود VB.NET
```
empDataSet.Tables.Add(EmployeeTable)
```

9. 4. التعامل مع DataColumn

تمثل هذه الفئة عمود واحد في قاعدة البيانات ، وبمعنى آخر فإن مجموعة من ال DataColumns تشكل DataTable يحتوي على هيكل قاعدة البيانات ، المكونات الأساسية لهذه الفئة هي:

الخاصية	الوصف
AllowDBNull	لتحديد السماح بعدم احتواء بعض القيم في هذا العمود على القيمةNull
AutoIncrement AutoIncrementSeed AutoIncrementStep	تتيح لك هذه الخصائص الثلاث عمل خاصية ترقيم تلقائي كما اوضحناها في بداية دورسنا عن SQL Server ، وتستخدم لتحديد الخاصية وتحديد نقطة البداية ومقدار الزيادة مع كل مرة على الترتيب
Caption	اسم العمود
DataType	نوع البيانات
DefaultValue	القيمة الافتراضية لقيم الحقول في هذا العمود
Table	تحديد ال DataTable الذي يتبع له هذا العمود
Unique	حدد كون هذه القيمة غير قابلة للتكرار Primary key

الجدول 17. 4. بعض خصائص الفئة DataColumn.

انشاء DataColumns:

لنفترض انشاء قاعدة بيانات تحتوي على الرقم الذي هو مفتاح رئيسي وايضاً ترقيم تلقائي، أما الاسم والعمر فهما لا يحتويان على أي قيم مميزة ، سيكون هذا هو شكل الإعمدة في هذا الجدول:

كود C#

```
DataColumn EmpIDColumn = new DataColumn("ID", typeof(int));
EmpIDColumn.Caption = "Employee ID";
EmpIDColumn.ReadOnly = true;
EmpIDColumn.AllowDBNull = false;
EmpIDColumn.AutoIncrement = true;
EmpIDColumn.AutoIncrementSeed = 1;
EmpIDColumn.AutoIncrementStep = 1;
EmpIDColumn.Unique = true;
DataColumn EmpNameColumn = new DataColumn("Name", typeof(string));
EmpNameColumn.Caption = "Employee Name";
DataColumn EmpAgeColumn = new DataColumn("Age", typeof(int));
EmpNameColumn.Caption = "Employee Age";
```

كود	VB.NET

```
Dim EmpIDColumn As New DataColumn("ID", GetType(Integer))
EmpIDColumn.Caption = "Employee ID"
EmpIDColumn.[ReadOnly] = True
EmpIDColumn.AllowDBNull = False
EmpIDColumn.AutoIncrement = True
EmpIDColumn.AutoIncrementSeed = 1
EmpIDColumn.AutoIncrementStep = 1
EmpIDColumn.Unique = True
Dim EmpNameColumn As New DataColumn("Name", GetType(String))
EmpNameColumn.Caption = "Employee Name"
Dim EmpAgeColumn As New DataColumn("Age", GetType(Integer))
EmpNameColumn.Caption = "Employee Age"
```

ومن ثم نضيفها إلى ال DataTable بالشكل التالي:

كود	C#

```
EmployeeTable.Columns.AddRange(new DataColumn[] { EmpIDColumn, EmpNameColumn,
EmpAgeColumn });
```

كود	VB.NET

```
EmployeeTable.Columns.AddRange(New DataColumn() {EmpIDColumn, EmpNameColumn,
EmpAgeColumn})
```

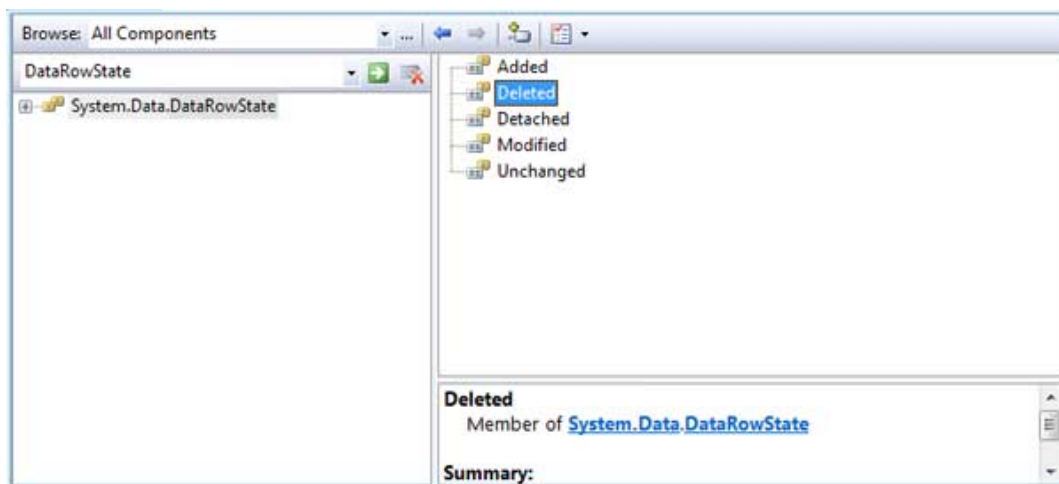
9 .5. التعامل مع DataRow

تختص هذه الفئة بالتعامل مع الصفوف ، المكونات الرئيسية لهذه الفئة هي:

الخاصية	الوصف
ItemArray	كافة بيانات هذا الصف على شكل مصفوفة array من الobjects
Table	الجدول الذي يحتوي على هذا الصف
AcceptChanges()	تطبيق التغييرات التي تمت على هذا الصف
RejectChanges()	الغاء التغييرات التي تمت على هذا الصف منذ آخر مرة تم استدعاء AcceptChanges() فيها

BeginEdit()	بدء عملية التعديل على هذا الصف
EndEdit()	انهاء عملية التعديل و حفظ التعديلات
CancelEdit()	الغاء عملية التعديل
Delete()	حذف هذا الصف عند استدعاء AcceptChanges()
IsNull()	لمعرفة قيمة عمود ما في هذا الصف وهل هي null ام لا
RowState	حالة الصف والتي يمكن قراءتها من ال enum المسمى DataRowState

الجدول 17. 5. بعض خصائص الفئة DataRow.

والذي يحتوي – DataRowState - على القيم التالية:

الصورة 17. 12. ال enum DataRowState.

انشاء DataRows

لا يمكن انشاء DataRow مباشرة ، مثل هذا السطر سوف ينشأ خطأ:

كود C#

```
DataRow r = new DataRow();
```

كود	VB.NET

```
Dim r As DataRow = new DataRow();
```

لكن لا بد من انشاء DataRow من نسخة من الجدول ، هذا المثال لانشاء نسخة من DataRow ووضع قيم بعض الصفوف فيها حسب الجدول الذي قمنا بعمله في البداية.

كود	C#

```
DataRow EmpRow = EmployeeTable.NewRow();
EmpRow["Name"] = "Ahmed Gamal";
EmpRow["Age"] = 22;
EmployeeTable.Rows.Add(EmpRow);
EmpRow = EmployeeTable.NewRow();
EmpRow[1] = "Ahmed Essawy";
EmpRow[2] = 23;
EmployeeTable.Rows.Add(EmpRow);
```

كود	VB.NET

```
Dim EmpRow As DataRow = EmployeeTable.NewRow()
EmpRow("Name") = "Ahmed Gamal"
EmpRow("Age") = 22
EmployeeTable.Rows.Add(EmpRow)
EmpRow = EmployeeTable.NewRow()
EmpRow(1) = "Ahmed Essawy"
EmpRow(2) = 23
EmployeeTable.Rows.Add(EmpRow)
```

لو لاحظت اننا لم نضع الحقل ID نظراً لإنه ترقيم تلقائي ، كما ان الحالة الأولى فيها الوصول للحقول بالاسم اما الثانية فبرقم ال Index.

- مثال شامل: معرفة كل الجداول في قاعدة البيانات DataSet ومعرفة الصفوف والأعمدة:

كود	C#

```
Text1.Text = "";
foreach (DataTable dt in empDataSet.Tables)
{
    Text1.Text += dt.TableName + ":\n\r";
    for (int curCol = 0; curCol < dt.Columns.Count; curCol++)
    {
        Text1.Text += (curCol + 1).ToString() + dt.Columns[curCol].ColumnName +
"\n\r";
    }
    Text1.Text = "Rows: \n\r";
    for (int curRow = 0; curRow < dt.Rows.Count; curRow++)
    {
        for (int curCol = 0; curCol < dt.Columns.Count; curCol++)
        {
            Text1.Text += dt.Rows[curRow][curCol].ToString() + " - ";
        }
    }
}
```

كود	VB.NET

```
Text1.Text = ""
For Each dt As DataTable In empDataSet.Tables
    Text1.Text += dt.TableName + ":" & Chr(10) & "" & Chr(13) & ""
    For curCol As Integer = 0 To dt.Columns.Count - 1
        Text1.Text += (curCol + 1).ToString() + dt.Columns(curCol).ColumnName +
"" & Chr(10) & "" & Chr(13) & ""
    Next
    Text1.Text = "Rows: " & Chr(10) & "" & Chr(13) & ""
    For curRow As Integer = 0 To dt.Rows.Count - 1
        For curCol As Integer = 0 To dt.Columns.Count - 1
            Text1.Text += dt.Rows(curRow)(curCol).ToString() + " - "
        Next
    Next
Next
```

6 .9. استخدام DataTableReader لقراءة البيانات من DataTable

كود C#

```
Text1.Text = "";
DataTableReader dtReader = dt.CreateDataReader();
while (dtReader.Read())
{
    for (int i = 0; i < dtReader.FieldCount; i++)
    {
        Text1.Text += dtReader.GetValue(i).ToString();
    }
}
```

كود VB.NET

```
Text1.Text = ""
Dim dtReader As DataTableReader = dt.CreateDataReader()
While dtReader.Read()
    For i As Integer = 0 To dtReader.FieldCount - 1
        Text1.Text += dtReader.GetValue(i).ToString()
    Next
End While
```

10. عمل Serializing إلى XML

كود C#

```
EmpDataSet.WriteXml("Employee.xml");
```

كود VB.NET

```
EmpDataSet.WriteXml("Employee.xml")
```

أو العكس:

كود C#

```
EmpDataSet.ReadXml("Employee.xml");
```

كود | VB.NET

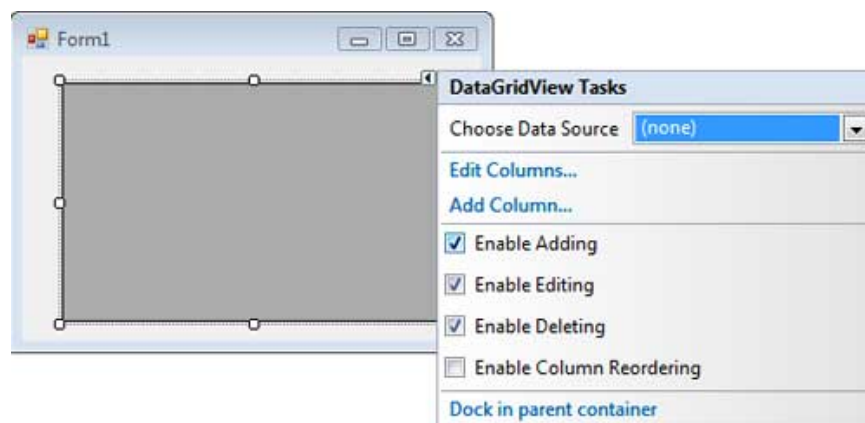
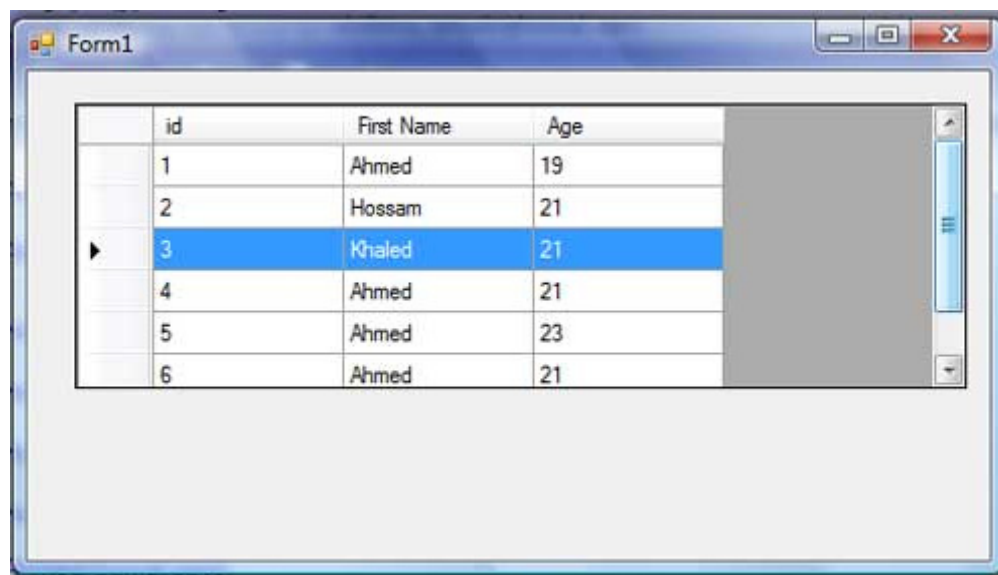
```vb
EmpDataSet.ReadXml("Employee.xml")
```

سيكون ناتج الملف Employee.xml بهذا الشكل تقريباً:

كود | XML

```xml
<?xml version="1.0" standalone="yes"?>
<Employee_XML>
  <Employee>
    <ID>1</ID>
    <Name>Ahmed Gamal</Name>
    <Age>22</Age>
  </Employee>
  <Employee>
    <ID>2</ID>
    <Name>Ahmed Essawy</Name>
    <Age>23</Age>
  </Employee>
</Employee_XML>
```

11. استخدام ال DataGrid

في مشروعنا الذي قمنا بانشاءه سابقاً ووضعنا فيه جدول الموظفين ، كل ما عليك هو سحب اداة DataGridView من ضمن الأدوات الموجودة بالشكل التالي:

الصورة 17. 13. اضافة الأداة DataGridView

كل ما علينا هو ربط قاعدة البيانات بالمصدر عن طريق Choose DataSource ، قم باختيار اضافة New DataSource ، وقم بتتبع المعالج ، اضافة DataBase ومن ثم نقوم باختيار قاعدة البيانات الخاصة بنا.

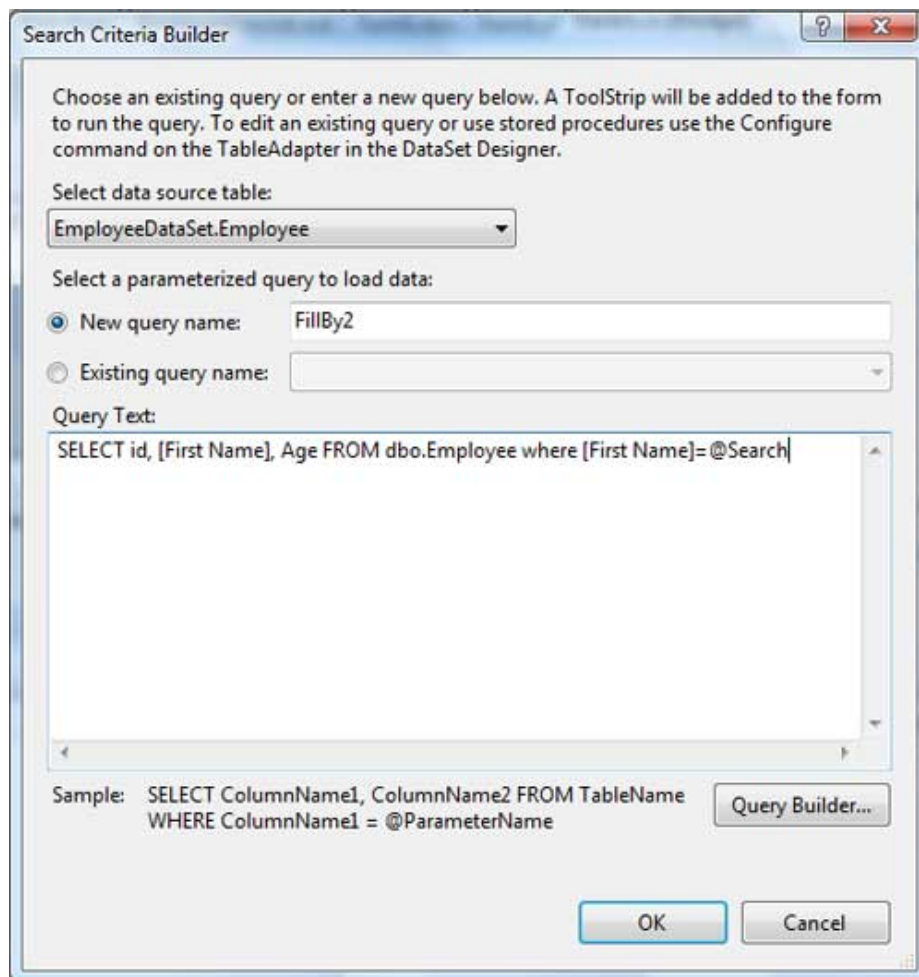
*** في العادة بدلاً من ابحث عن ال ConnectionString اقوم بعمل DataSource واستعرض ال ConnectionString، ثم اقوم بالغائها☺

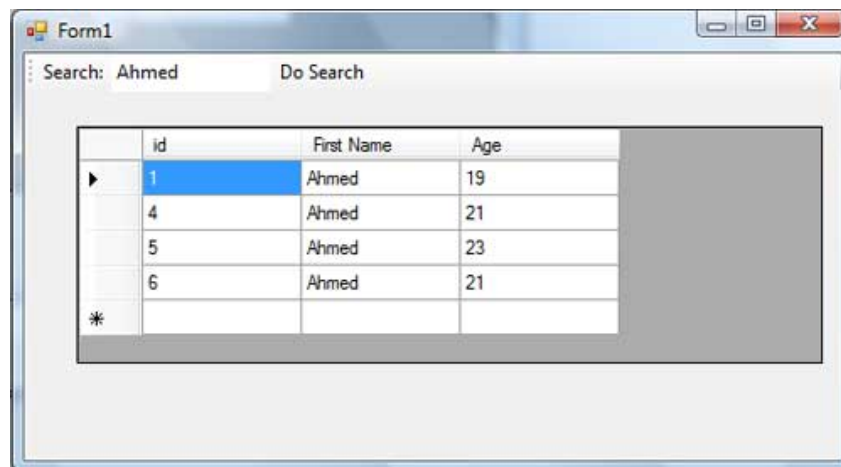
بعد انتهاء الشاشة الخاصة باختيار الداتا سورس ، تظهر لك شاشة تخبرك باختيار الجدول أو جملة الاستعلام التي تود لها أن تظهر في الجدول ، سنجرب اختيار جدول مباشرة وهو جدول الموظفين ، وفي مثال لاحق سنجرب اختيار جملة استعلام.

سيكون الناتج شيئاً مثل هذا:

الصورة 17. 14. اضافة الأداة DataGridView و تعيين مصدر البيانات Data Source.

بالطبع يمكنك اتاحة الفرصة للتعديل والاضافة من الجدول.

الآن سنجرب تعديل بسيط جداً ، سنطبق شرط للبحث بحيث لا يتم عرض البيانات سوى التي لها الاسم الأول = الأسم الأول الموجود في مربع نص ، من ال dataGrid اختر Add Query ومن ثم اكتب جملة استعلام كالتالي:

الصورة 17. 15. أضافة شروط على البيانات المعروضة.

ستلاحظ ظهور شاشة في الأعلى بالشكل التالي ، جرب كتابة الاسم حتى النهاية وشاهد النتائج:

الصورة 17. 16. نتائج العرض.

12. استخدام ال DataAdapter

ببساطة شديدة، قم بتعريف DataAdapter كما تعلمنا سابقاً:

كود C#

```
string connectionString = @"Data Source=AHMED-PC\SQLEXPRESS;Initial
Catalog=Employee;Integrated Security=True;Pooling=False";
DataSet ds = new DataSet("Employee");
SqlDataAdapter dAdapt = new SqlDataAdapter("Select * From Employee_info",
connectionString);
dAdapt.Fill(ds, "Employee_info");
```

كود VB.NET

```
Dim connectionString As String = "Data Source=AHMED-PC\SQLEXPRESS;Initial
Catalog=Employee;Integrated Security=True;Pooling=False"
Dim ds As New DataSet("Employee")
Dim dAdapt As New SqlDataAdapter("Select * From Employee_info",
connectionString)
dAdapt.Fill(ds, "Employee_info")
```

الباب

18

LINQ

سنعود مرة أخرى مع LINQ، هذه المرة سنهتم بتعاملها مع قواعد البيانات المختلفة .

1. مقدمة

بعد ان تعرفنا على بعض النقاط في عالم ADO.net، جاء الدور لنتعرف على تقنية Language Integrated Query او ما يعرف اختصاراً باسم LINQ، وهي تقنية جديدة من مايكروسوفت تهدف كما اسلفنا في درس سابق إلى بناء استعلامات قوية يمكن التحكم فيها من خلال بيئة فيجوال ستوديو ، اضف إلى ذلك توحيد جملة الاستعلام مهما تكن نوعية مصدر قواعد البيانات الذي نتعامل معه سواء كان قاعدة بيانات أو XML File أو ملف نصي او ملف اكسيل أو خلافه.

لل LINQ ثلاث انواع رئيسية سنحاول التعرف على أجزائها هي:

1- LINQ to SQL
2- LINQ to XML
3- LINQ to Objects

أبسط أنواع الاستعلام هي الاستعلام مباشرة إلى متغير من نوع Array ، لنفترض المثال الذي شرحناه في مقدمتنا إلى LINQ قبل عدة دروس بالشكل التالي:

كود | C#

```
from d in developers
where d.Language == "C#"
select d.Name;
```

هذه هي نفسها جملة الاستعلام التي كنا نكتبها بالشكل التالي:

كود | SQL

```
SELECT name FROM developers WHERE language="C#"
```

ولكن الآن لم نعد نحتاج لوجود قاعدة بيانات لتنفيذ جملة الاستعلام هذه عليها ، بل صار بامكاننا تطبيقها على مصفوفة بالشكل التالي:

C# كود

```
public class Developer
{
    public string Name;
    public string Language;
}
Developer[] developers = new Developer[]
{  new Developer { Name = " Ahmed", Language="C#" },
    new Developer { Name="Khaled", Language="Java" },
    new Developer { Name="Ali", Language="C++"}
};
```

VB.NET كود

```
Public Class Developer
    Public Name As String
    Public Language As String
End Class
Dim developers As Developer()  = new With { New Developer { .Name = " Ahmed",
.Language="C#" }, New Developer { .Name="Khaled", .Language="Java", New
Developer { .Name="Ali", .Languag="C++"} }
```

ومن ثم نكتب جملة استعلام بسيطة بالشكل التالي :

C# كود

```
IEnumerable<string> LINQresult = from d in developers
                                 where d.Language == "C#"
                                 select d.Name;
```

VB.NET كود

```
Dim LINQresult As IEnumerable(Of String) = From d In developers _
                                           Where (d.Language = "C#") _
                                           Select d.Name
```

الآن اصبح بامكانك بكل بساطة طباعة الناتج بالشكل التالي:

C# كود

```
foreach (string s in LINQresult)
{
    MessageBox.Show(s);
}
```

VB.NET كود

```
For Each s As String In LINQresult
     MessageBox.Show(s)
Next
```

كما تلاحظ يمكنك استخدام اي معامل من المعاملات السابقة او عدمه ، فجملة كالتالي صحيحة ما دمت لا تحتاج الشرط:

كود C#
```
IEnumerable<string> LINQresult = from d in developers
                                 select d.Name;
```

كود VB.NET
```
Dim LINQresult As IEnumerable(Of String) = From d In developers() _
                                          Select d.Name
```

كما يمكنك ايضاً استخدام دالة .net عادية ضمن الشرط مثل دالة الطول بالشكل التالي:

كود C#
```
IEnumerable<string> LINQresult = from d in developers
                                 where d.Name.Length > 3
                                 orderby d.Name
                                 select d.Name;
```

كود VB.NET
```
Dim LINQresult As IEnumerable(Of String) = From d In developers _
                                            Where (d.Name.Length > 3) _
                                            Order By (d.Name) _
                                            Select d.Name
```

ترتيب جمل الاستعلام

تعودنا في جمل الاستعلام على ترتيب Where From Select، ولكننا في LINQنستخدم Select لتكون هي الأخيرة دائماً ، السبب بكل بساطة هو ما يعرف باسم IntelliSenseحيث اننا نود ان نعتمد على الفيجوال ستوديو ليظهر لنا الأعمدة التي يمكنها الظهور لنا ، فيما لو كتبنا Select قبل From فلن يستطيع الفيجوال ستوديو مساعدتك في تحديد الخيارات الموجودة لإنه لا يعرف اي جدول تريد أن تقرأ منه بعد.

جملة الاستعلام الكاملة:

كود

```
from id in source
{
from id in source /
Join id in source on expr equals expr [into id] |
Let id = expr |
Where condition |
Orderby ordering, ordering, ... [Ascending | Descending]
}
```

ضع هذه الصيغة العامة في بالك فقط ، سنتعرف عليها خلال عملنا في هذه الدروس إن شاء الله ...

2. دوال LINQ

تحتوي LINQ على 51 دالة ووظيفة مختلفة ، سنحاول الآن التعرف على معاني وشروحات هذه المجموعة من الدوال والتي سنستخدم بعضاً منها خلال دروسنا .

الدالة	الوصف
Aggregate	للتطبيق المتكرر للعمليات الحسابية ، مثلاً جمع عدة أرقام أو ضربهم أو أي عملية حسابية ، المثال التالي : `int product = ints.Aggregate(1, (a, b) => a * b);` حيث أن ints هي مصفوفة من الأرقام .
All	لمعرفة مدى تطبيق شرط على الكل ، لنفترض مثلاً أنك تود معرفة هل كل الموظفين متزوجين أم لا : `bool isMarried = employees.All(emp => emp.statues == 1);`
Any	على عكس السابقة ، فهي تعيد قيمة صحيحة في حالة كون واحد فقط على الأقل من المجموعة يحقق الشرط ، هذا المثال : `bool isMarried=employees.Any(emp = > employees.statues==1);`
Average	المتوسط لمجموعة من القيم ، هذا المثال : `double avg = ints.Average();` حيث أن ints هي مجموعة من الأرقام .

Cast	لتحويل مجموعة من العناصر إلى نوع محدد ، هذا المثال : `var newString = ints.Cast<string>();` حيث أن ints هي مجموعة من الأرقام .
Concat	لدمج عنصرين ، فمثلاً لو كان لدينا query1 له نتيجة و query2 له نتيجة أخرى ونود دمجهم يمكن استخدام الأمر التالي : `var result = query1.Concat(query2);`
Contains	تستخدم للبحث داخل مصفوفة ، مثلاً للبحث عن موظف اسمه محمد نكتب السطر التالي : `bool find=employees.Contains(new employee("Mohammed");` وهي مشابهه لعمل ذلك عن طريق Query عادي .
Count	يقوم بعد نتائج الإستعلام أو عدد عناصر المصفوفة أو أياً ما يكن : `int count = employees.Count();`
DefualtIfEmpty	إذا كانت هناك قيمة فارغة فيتم وضع قيمة افتراضية لها : `var result = ints.DefaultIfEmpty(100);` لو لم يتم تحديدها فسيتم وضع القيمة الافتراضية لنوع البيانات ، مثلاً صفر للأرقام و "" للنصوص و null لأي Object.
Distinct	يقوم باعادة البيانات بدون أي تكرارات ، مثلاً : `var pure = query.Distinct();`
ElementAt	يقوم باعادة القيمة في مكان معين ، مثلاً : `Employee newemp = query.ElementAt(4);`
ElementAtOrDefualt	لو لاحظت في المثال السابق أنك قد تكتب رقم 4 في حين انت نتيجة الإستعلام لم تعد أربع صفوف أصلاً ، لذا تخبرنا الجملة التالية أنه في حالة كون الرقم غير موجود يتم الاستعاضة عنه بالقيم الافتراضية للفئة أو لنوع البيانات : `Employee newemp = query.ElementAtOrDefualt(4);`
Empty	يقوم بانتاج مجموعة فارغة من العناصر ، بالشكل التالي مثلاً : `var newresult = System.Query.Sequence.Empty<employee>();`
EqualAll	المقارنة بين نتيجتي استعلام والتأكد من أنهما متطابقتان : `bool iseuql = query1.EqualAll(query2);`
Except	مقارنة مجموعتين من العناصر واسترجاع القيم الموجودة في المجموعة الأولى دون أن تكون في المجموعة الثانية : `var result = query1.Except(query2);` الناتج سيكون بدون تكرارات ...

First	يقوم بإعادة أول عنصر في المجموعة :
	`Employee firstone = employees.First();`
	*** لا بد أن لا تكون مجموعة العناصر هذه فارغة .
	كما يمكن أيضاً أن يعيد أول عنصر في استعلام فرعي ، فلإعادة أول موظف متزوج :
	`Employee firstone = employees.First(emp => emp.ismarried == 1);`

FirstOrDefault	مثل السابق ما عدا أنه في حالة كون المجموعة فارغة أو الاستعلام الفرعي لم يعد بنتيجة أن يعيد القيمة الافتراضية : `Employee firstone = employees.FirstOrDefault(emp => emp.ismarried == 1);`
Fold	مثل ال Aggergate
GroupBy	تجميع العناصر حسب التحديد ، مثل GroupBy الطبيعية في ال Sql Statement ، هذا المثال : `var newGroup = employees.GroupBy(emp => emp.Country);` ويسمى key ويمكن الوصول له عبر هذه الخاصية .
GroupJoin	عملية ال Join .
Intersect	تقاطع مجموعين ، العناصر الموجودة في كلا المجموعتين فقط : `var inter = query1.Intersect(query2);`
Join	عملية ال inner join في Sql Statements .

Last	مثل First ولكنها تعيد آخر عنصر :
	`Employee lastone=employees.Last();`
	`Employee lastone=employees.Last(emp => emp.Country==”Egypt”);`

LastOrDefault	مثل السابق ما عدا أنه في حالة كون المجموعة فارغة أو الاستعلام الفرعي لم يعد بنتيجة أن يعيد القيمة الافتراضية : `Employee firstone = employees.LastOrDefault(emp => emp.ismarried == 1);`
LongCount	مثل Count ولكن الناتج يكون من النوع Long هذا المثال : `long empCount = employees.LongCount();`
Max	أكبر عنصر قيمة في المجموعة : `int num = ints.Max();` ويمكن تحديد طريقة لمعرفة المقارنات بين الفئات كما تعلمنا في دروس سابقة

Min	اصغر عنصر قيمة في المجموعة : `int num = ints.Min();` ويمكن تحديد طريقة لمعرفة المقارنات بين الفئات كما تعلمنا في دروس سابقة
OfType	استخراج العناصر من نوع معين فقط ، لاستخراج العناصر ال int فقط مثلاً نكتب : `var newResult = list.OfType<int>();`
OrderyBy	لترتيب النتائج حسب طريقة معينة (تصاعدياً) هذا الكود كمثال : `var orderlist = employees.OrderBy(emp => emp.Age);`
OrderyByDescending	الترتيب ولكن بصورة تنازلية ، هذا المثال : `var orderlist = employees.OrderyByDescending(emp => emp.ID);`
Range	توليد مجموعة من الأرقام الصحيحة بين حدين معينين : `var newlist = System.Query.Sequence.Range(1, 5);`
Repeat	تكرار رقم معين عدة مرات ، مثلاً لتكرار رقم "1" خمس مرات : `var list = System.Query.Sequence.Repeat(1, 5);`
Reverse	عكس ترتيب أي مجموعة : `var revList = list.Reverse();`

Select	جملة البحث العادية، يمكن أن يستعلم عن الكل:
	`var newSelect = list.Select(emp => emp);`
	أو أجزاء معينة فقط
	`var newSelect = employees.Select(emp => new { emp.Name, emp.Age });`

SelectMany	في حالة استخدامنا ل Group فإن الناتج لن يكون صفاً واحداً فقط ، لذلك نستخدم SelectMany بالشكل التالي : `var newSelect = groupedlist.Select(l => l);`
Single	اعادة قيمة واحدة فقط بشرط ألا تحتوي المجموعة على غيره : `var result = query.Single();`
SingleOrDefault	نفس السابق عدا أنه يعيد القيمة الافتراضية في حالة عدم وجود أي عناصر: `var result = query.SingleOrDefault();`
Skip	تجاوز عدد من العناصر واعادة الباقي ، مثلاً لتجاوز أول 10 عناصر واعادة البقية : `var newResult = query1.Skip(10);`

SkipWhile	مثل السابق ولكنه يتجاهل العناصر حتى تحقق شرط معين وليكن مثلاً تجاوز رقم ما حد ال 100 : `var newResult = query1.SkipWhile(x => x < 100);`
Sum	جمع قيم مجموعة : `Double sum = ints.Sum();`
Take	يقوم باسترجاع عناصر بعدد معين من البداية : `var top = ints.Take(3);`
TakeWhile	استرجاع عناصر حتى حدوث شرط ما ، مثلاً الاسترجاع حتى ايجاد رقم اكبر من 100 : `var top = ints.TakeWhile(x => x < 100);`
ThenBy	لتحديد الأولويات ، مثلاً في الكود الخاص بالترتيب للترتيب بناء على العمر ثم على المرتب مثلاً : `var orderemp=employees.OrderBy(emp => emp.Age ThenBy emp => emp.Salary);`
ThenByDesecnding	نفس السابق ما عدا أن الترتيب هذه المرة سوف يكون تنازلياً : `var orderemp=employees.OrderBy(emp => emp.Age ThenByDesecnding emp => emp.Salary);`
ToArray	تحويل إلى مصفوفة : `Employee[] emps = query.ToArray();`
ToDictonary	نفس السابق ولكن مع تخزينها في مصفوفة ثنائية الأبعاد .
ToList	تخزينها في كائن List<T>
ToLookup	تخزين الناتج في كائن Lookup<K, V>
ToSequence	تحويل الناتج إلى Sequence متتابع حيث يحل مشكلة التعارضات في الأسماء بين الدوال وأسماء الحقول .
Union	عملية الاتحاد الرياضية، حيث يحتوي الناتج على مجموعة تشمل كل عناصر المجموعتين: `var result = query1.Union(query2);`
Where	جملة البحث بشرط العادية .

الجدول 18. 1. الكلمات المفتاحية لتقنية ال LINQ.

ملاحظة

استفدت كثيراً في الجزء السابق من كتاب (إبدأ LINQ) يمكنك تنزيله من هذا الرابط

http://www.vb4arab.com/vb/uploaded/2730/11202261862.pdf

تذكير سريع : المتغيرات غير المعرفة

احياناً تقوم بعمل جملة استعلام يكون الناتج فيها فئة لم تقم بتعريفها من قبل ، هذه هي واحدة من فوائد تقنية Implicitly Typed Local Variables التي تم شرحها سابقاً ، هذه الجملة كمثال:

كود	C#

```
var subset = from i in numbers
             where i < 10
             select i;
```

* تذكير آخر : استخدام Lambda Expressions مع LINQ :

تذكر أن باستطاعات استخدام تعبيرات لامبادا Lambda Expressions بدلاً من كتابة استعلام LINQ بالطريقة العادية.

3. LINQ To DataSet

ال DataSet كما اسلفنا تشكل صورة أو نسخة من قاعدة البيانات في الذاكرة ، لذا فهي بكل بساطة نقطة جيدة لنستخدم عليها استعلامات LINQ الخاصة بنا.

فمثلاً لنجرب ربط قاعدة بيانات باستخدام DataSet ومن ثم ربطها بـ DataTable والحصول على بعض البيانات منها :

C# كود

```
DataSet ds = LoadDataSetSomeway();
DataTable employees = ds.Tables["Employee"];
var query =
    from emp in employees.AsEnumerable()
    where emp.Field<long>("Age").Year >= 40
    select emp;
```

VB.NET كود

```
Dim ds As DataSet = LoadDataSetSomeway()
Dim employees As DataTable = ds.Tables("Employee")

Dim query = From emp In employees.AsEnumerable() _
            Where emp.Field(Of Long)("Age").Year >= 40 _
            Select emp
```

4. LINQ To XML

أصبحت XML الآن أحد أكثر مخازن البيانات شيوعاً ، و LINQ تتعامل مع XML مثل تعاملها مع أي نوع آخر من قواعد البيانات ، اضافة لبعض الخصائص الإضافية التي تختص ب XML فقط والتي سنتعرف عليها سوية .

فمثلاً لو افترضنا ملف XML يحتوي على بعض المنتجات بالشكل التالي :

كود XML

```xml
<?xml version="1.0"?>
<Items>
  <Item Number="122">
    <ItemName>Item1</ItemName>
    <Quantity>100</Quantity>
    <Price>23</Price>
  </Item>
  <Item Number="123">
    <ItemName>Item2</ItemName>
    <Quantity>10</Quantity>
    <Price>14.5</Price>
  </Item>
  <Item Number="124">
    <ItemName>Item3</ItemName>
    <Quantity>31</Quantity>
    <Price>1000</Price>
  </Item>
  <Item Number="125">
    <ItemName>Item4</ItemName>
    <Quantity>22</Quantity>
    <Price>97</Price>
  </Item>
</Items>
```

بداية سنقوم بعمل Load لملف ال XML بالشكل التالي :

كود C#

```csharp
XElement purchaseOrder = XElement.Load("Items.xml", LoadOptions.SetBaseUri |
LoadOptions.SetLineInfo);
```

كود VB.NET

```vb
Dim purchaseOrder As XElement = XElement.Load("Items.xml",
LoadOptions.SetBaseUri Or LoadOptions.SetLineInfo)
```

الآن ، لو افترضنا أننا نود استعراض مجموعة الأصناف الموجودة لدينا ، سيكون الكود الذي نكتبه بالشكل التالي :

كود C#

```csharp
var newSearch  = from item in Items.<Item>
                 select item.@Number;
```

كود	VB.NET

```
Dim newSearch = From item In Items.<Item>() _
                Select item.@Number
```

وللعثور على المجموعة التي سعرها أكبر من 100 دولار مثلاً ؟

كود	C#

```
var newSearch  = from item In Items.<Item>
                 where item.<Price>.Value > 100
                 select item.@Number
```

كود	VB.NET

```
Dim newSearch = From item In Items.<Item>() _
                Where (item.<Price>.Value > 100) _
                Select item.@Number
```

5. LINQ To SQL

فئة من فئات LINQ موجهة خصيصاً لدعم SQL Server ، في العادة هي الأكثر استخداما وسنحاول التعرف عليها سريعاً في هذا الدرس.

البحث والاستعلام

تعرفنا عليه سابقاً ، هذا المثال مثلاً لاعادة الاسم وتاريخ التخرج على شكل فئة جديدة للأشخاص الذين يزيد عمرهم عن 30 سنة ويقطنون في مصر:

كود	C#

```
var query = from c in Employee
            where c.Age > 30 && c.Country == "Egypt"
            select new { c.Name, c.GraduationDate };
```

كود	VB.NET

```
Dim query = From c In Employee _
            Where (c.Age > 30) AndAlso c.Country = "Egypt" _
            Select New With {c.Name, c.GraduationDate}
```

استدعاء Stored Procedures

كل ما عليك هو فقط تعريف دالة تشابه Procedure لها نفس ال Attributes ، وهي ما يقوم بها الفيجوال ستوديو افتراضياً ، الآن كل ما عليك هو كتابة أمر بهذا الشكل (استعلام يستقبل متغير واحد) :

كود C#

```
var c = db.ProceduresName("Egypt");
```

كود VB.NET

```
Dim c = db.ProceduresName("Egypt")
```

إذا كان هناك جملة استعلام ستستخدمها لعدة مرات فليس من المنطقي جعل نظام قواعد البيانات يتعرف من جديد على الاستعلام كل مرة ، لذا توفر لنا نظم قواعد البيانات الحل لهذه النقطة ، أيضاً LINQ تقدم لنا ذات النظرية عبر ما يسمى باسم Compiled Query.

فكرة ال Compiled Query ببساطة تعتمد على استخدام دالة CompiledQuery.Compile لعمل Compile لجملة الاستعلام ، الشكل التالي كمثال:

كود C#

```
var c = CompiledQuery.Compile((DataContext context, string CountryName, int
age) =>
                         from c in Employee
                         where c.Age > age && c.Country == CountryName
                         select new { c.CustomerID, c.CompanyName, c.City });
```

كود VB.NET

```
Dim c = CompiledQuery.Compile(Function(context As DataContext, CountryName As
String, age As Integer) From c In Employee _
    Where c.Age > age AndAlso c.Country = CountryName _
    Select New With {c.CustomerID, c.CompanyName, c.City})
```

والآن يمكنك تجربة استخدامها لأكثر من مرة بتغيير البارميتر فقط ، والبحث في الاستعلام الذي تم عمل Compile له:

C# كود

```
foreach (var row in query(dc, "Egypt", 22))
{
}
foreach (var row in query(dc, "USA", 28))
{
}
foreach (var row in query(dc, "Saudi", 38))
{
}
```

VB.NET كود

```
For Each row As var In query(dc, "Egypt", 22)
Next
For Each row As var In query(dc, "USA", 28)
Next
For Each row As var In query(dc, "Saudi", 38)
Next
```

استخدام دوال في جملة الاستعلام

لو فرضنا جملة الاستعلام التالية:

SQL كود

```
SELECT SUM( emp.salary ) AS TotalSalary FROM Employee emp JOIN Department dem
ON emp.DepartmentID = dep.ID GROUP BY dep.ID
```

نستطيع كتابتها بشكل LINQ بالشكل التالي:

C# كود

```
var c = from emp in db.Employee
        join dep in db.Department
        on emp.DepartmentID equals dep.ID
        into TotalSalary
        select TotalSalary.Sum(emp => emp.Salary);
```

VB.NET كود

```
Dim c = From emp In db.Employee() _
        Join dep In db.Department _
        On emp.DepartmentID Equals dep.ID _
        Into(TotalSalary) _
        Select TotalSalary.Sum( emp => emp.Salary )
```

استخدام جمل الاستعلام داخل LINQ

قد تجد نفسك مضطراً احياناً لاستخدام واحدة من مميزات جمل الاستعلام التقليدية مثل الأمر PIVOT، على كل هذه هي الصيغة العام لتنفيذ جملة SQL داخل ال LINQ :

-مثال منقول - :

كود C#

```
var query = db.ExecuteQuery<EmployeeInfo>(@" With EmployeeHierarchy
(EmployeeID, LastName, FirstName, ReportsTo, HierachyLevel)
AS
( SELECT EmployeeID, LastName, ForstName, ReportsTo, 1 as HierarchyLevel
  FROM Employees
  WHERE ReportsTo IS NULL
  UNION ALL
        SELECT e.EmployeeID, e.LastName, e.FirstName, e.ReportsTo,
eh.HierarchyLevel + 1 AS HierarchyLevel
        FROM Emplyees e
        INNER JOIN EmployeeHierarchy eh
        ON e.ReportsTo = eh.EmplyeeID
)
SELECT *
FROM EmployeeHierarchy
ORDER BY HierarvhyLevel, LastName, FirstName");
```

كود VB.NET

```
Dim query = db.ExecuteQuery(Of EmployeeInfo)(" With EmployeeHierarchy
(EmployeeID, LastName, FirstName, ReportsTo, HierachyLevel)
AS
( SELECT EmployeeID, LastName, ForstName, ReportsTo, 1 as HierarchyLevel
  FROM Employees
  WHERE ReportsTo IS NULL
  UNION ALL
        SELECT e.EmployeeID, e.LastName, e.FirstName, e.ReportsTo,
eh.HierarchyLevel + 1 AS HierarchyLevel
        FROM Emplyees e
        INNER JOIN EmployeeHierarchy eh
        ON e.ReportsTo = eh.EmplyeeID
)
SELECT *
FROM EmployeeHierarchy
ORDER BY HierarvhyLevel, LastName, FirstName")
```

استخدام Inseret في LINQ

كود C#

```
var NewEmp = new Employee { FirstName = "Ahmed", Age = 25, Country="Egypt"};
db.Employee.Add( NewEmp );
```

كود VB.NET

```
Dim NewEmp = New With {.FirstName = "Ahmed", .Age = 25, .Country = "Egypt"}
db.Employee.Add(NewEmp)
```

في المثال السابق قمنا باضافة موظف جديد باسم أحمد وبعمر 25 سنة ومن مصر.

استخدام Update

كود C#

```
var EditEmp = db.Employee.Single (c => c.Country == "Egypt");
EditEmp.Age = 30;
```

كود VB.NET

```
Dim EditEmp = db.Employee.Single(Function(c) c.Country = "Egypt")
EditEmp.Age = 30
```

في المثال السابق قمنا بالتعديل لعمر كل الاشخاص من مصر ليصبح 30 سنة.

استخدام Delete

كود C#

```
var deleteEmp = db.Employee.Single(c => c.Age > 50);
db.Employee.Remove(deleteEmp);
```

كود VB.NET

```
Dim deleteEmp = db.Employee.Single(Function(c) c.Age > 50)
db.Employee.Remove(deleteEmp)
```

في المثال السابق قمنا بحذف الموظفين الذين تزيد اعمارهم عن 50 سنة.

6. SubmitChanges

بعد اي عملية للاضافة أو للحذف أو التعديل لا بد من استدعاء الدالة db.SubmitChanges() لتفعيل التغييرات في قاعدة البيانات.

التحديث بالتزامن في قاعدة البيانات

* ال LINQ تتعامل بالوضع المنفصل وليس المتصل ، لذا ستجد له الكائن ChangeConflictExeption والذي ينتج عن تعارض عمليات التعديل على قاعدة البيانات بين عدة مستخدمين ، يمكن استعراض رسالة الخطأ بالشكل التالي:

كود | C#

```
try
{
    Db2.SubmitChanges();
    break;
}
catch (ChangeConflictExeption ex)
{
    MessageBox.Show(ex.Message);
    Db2.Refresh(somequery, RefreshMode.KeepChanges);
}
```

كود | VB.NET

```
Try
    Db2.SubmitChanges()
    Exit Try
Catch ex As ChangeConflictExeption
    MessageBox.Show(ex.Message)
    Db2.Refresh(somequery, RefreshMode.KeepChanges)
End Try
```

هنا يحاول الاستعلام التنفيذ ، لو لم يستطع يظهر رسالة الخطأ ويحاول مع أمر Refresh مرة أخرى.

يمكن ايضاً في الأمر SubmitChanges تحديد سلوك البرنامج في حالة حدوث تعارض مثل الشكل التالي:

كود	C#

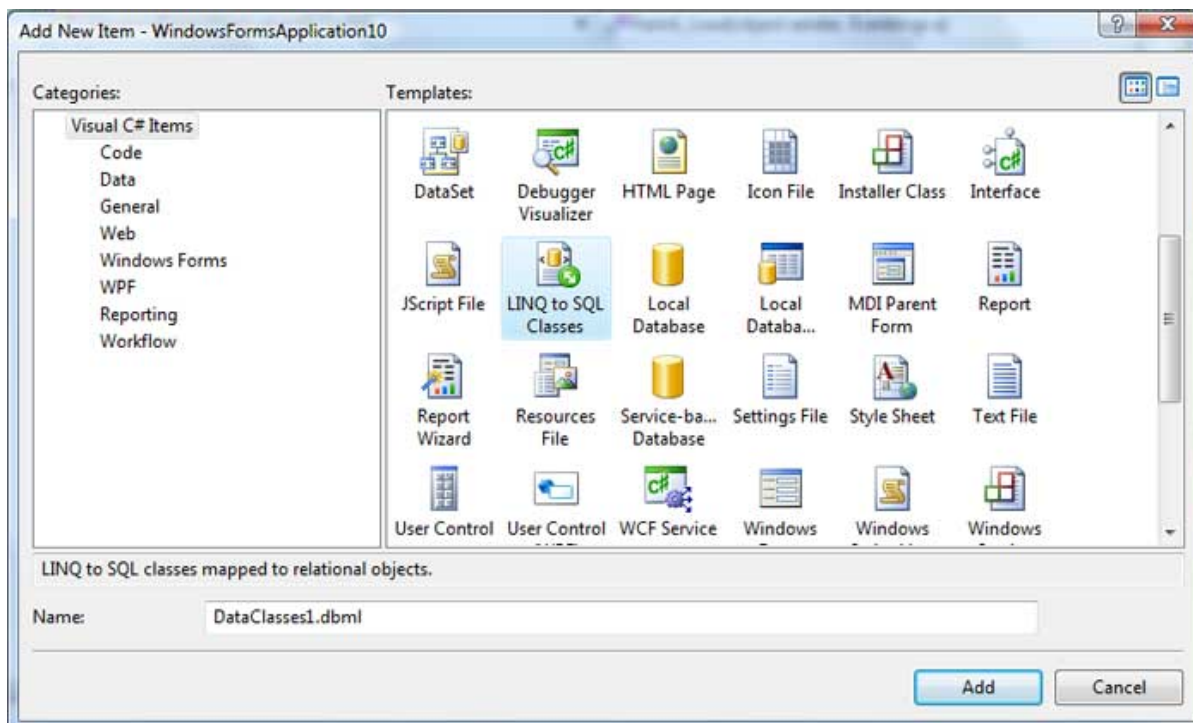
```
Db.SubmitChanges(ConflictMode.FailOnFirstConflict);
Db.SubmitChanges(ConflictMode.ContinueOnConflict);
```

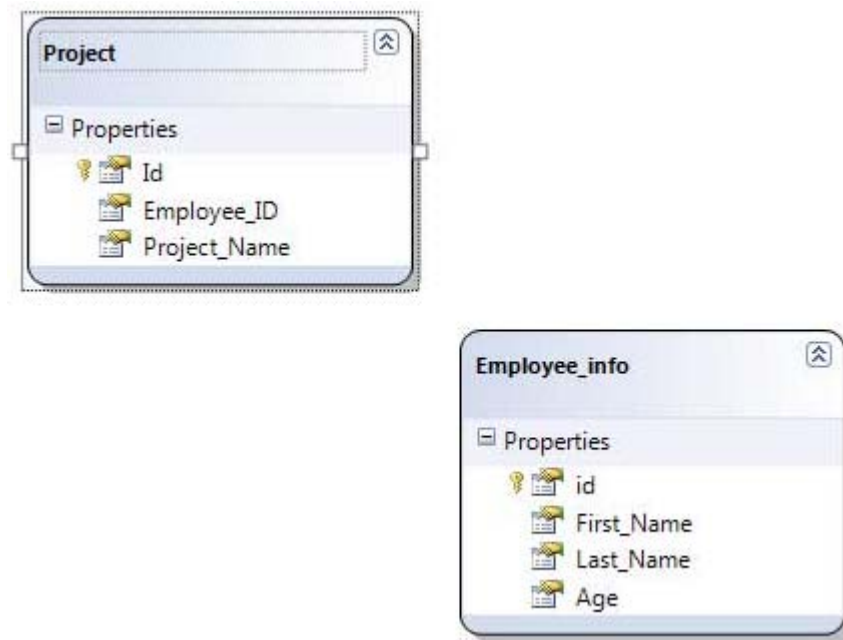
كود	VB.NET

```
Db.SubmitChanges(ConflictMode.FailOnFirstConflict)
Db.SubmitChanges(ConflictMode.ContinueOnConflict)
```

وذلك من الفئة ConflictMode والتي لا تحتوي سوى على هذين العنصرين.

7. انشاء فئات Linq To SQL من خلال الفيجوال ستوديو

يمكنك الفيجوال ستوديو 2008 من انشاء LINQ TO SQL Classes يمكنك التحكم بها لاحقاً ، من الشاشة الأولى:

الصورة 18. 1. نتائج العرض.

بعد اختياره ، يمكنك انشاء علاقات وجداول جديدة او سحب قاعدة بيانات من الServer Explorer، لذا قم بفتح قاعدة البيانات الخاصة بنا وقم بالبدء بسحب الجداول، ستجد الشكل العام التالي:

الصورة 18. 2. جداول قاعدة البيانات.

الآن بكل بساطة ستلاحظ ان الفيجوال ستوديو قام بمساعدتك في انشاءData Access Layer - لو كنت تتذكرها - ، ستجد تعريفات للدوال التالية مثلاً:

كود | **C#**

```
partial void InsertEmployee_info(Employee_info instance);
partial void UpdateEmployee_info(Employee_info instance);
partial void DeleteEmployee_info(Employee_info instance);
```

كود	VB.NET

```
Partial Private Sub InsertEmployee_info(ByVal instance As Employee_info)

End Sub
Partial Private Sub UpdateEmployee_info(ByVal instance As Employee_info)

End Sub
Partial Private Sub DeleteEmployee_info(ByVal instance As Employee_info)

End Sub
```

قبل أن أنهي الفصل الخاص ب LINQ، لو كنت تود معرفة المزيد عن LINQ فكتاب المهندس محمد سامر ربما يكون مناسباً لك كبداية جيدة في هذا المجال ، تجده على هذا الرابط :

رابط

http://www.vbcoffee.net/SamerSelo/linqcourseebook.pdf

الباب

19

WWF

منذ الاصدار net 3.0. تم اضافة ما يسمى بـWindows Workflow Foundation ويعرف اختصاراً WWF، وهي مجموعة من المهام والدوال API تسمح لك بادارة ومراقبة وتنفيذ الـ workflow او سير العمليات إن صحت الترجمة .

طبعاً تعد هذه الخاصية واحدة من انفع الخصائص لمهندسي البرمجيات التي تمت اضافتها داخل ال viusal studio حيث اصبح بالامكان دمج ال workflows مع الكود أو التنفيذ الفعلي مباشرة.

ال WF او ال Workflow هي مجموعة العمليات التي يتم تشكيل البرنامج بناء عليها ، حيث يتكون البرنامج من مجموعة مرتبطة مع بعضها البعض من ال Business Process والتي تحتوي بدورها على مجموعة من المهام ذات الصلة بينها وبين بعضها البعض والتي تعمل سوية.

مثال

خدمة الصيانة لمنتج : يحتوي هذا البرنامج على طلب الايصال وادخال رقمه وعمل Check على قاعدة البيانات للتأكد من وجوده ثم تقديم خدمة الصيانة ثم طباعة ايصال صيانة.

هذه العملية كلها تسمى Business Process والعناصر الداخلية تسمى tasks والبرنامج يتكون من عدة Business Process.

هناك العديد من البرامج التي تقدم خدمة اداة ال Workflow، ولكن الميزة هنا انها مرتبطة ارتباطاً وثيقاً بالكود بحيث يمكن عمل trace لكل واحدة على حدة وتتبع الأخطاء إن وجدت ، اضف إلى ذلك امكانيه التعديل لاحقاً في الجزئيات الكبرى بسهولة ويسر.

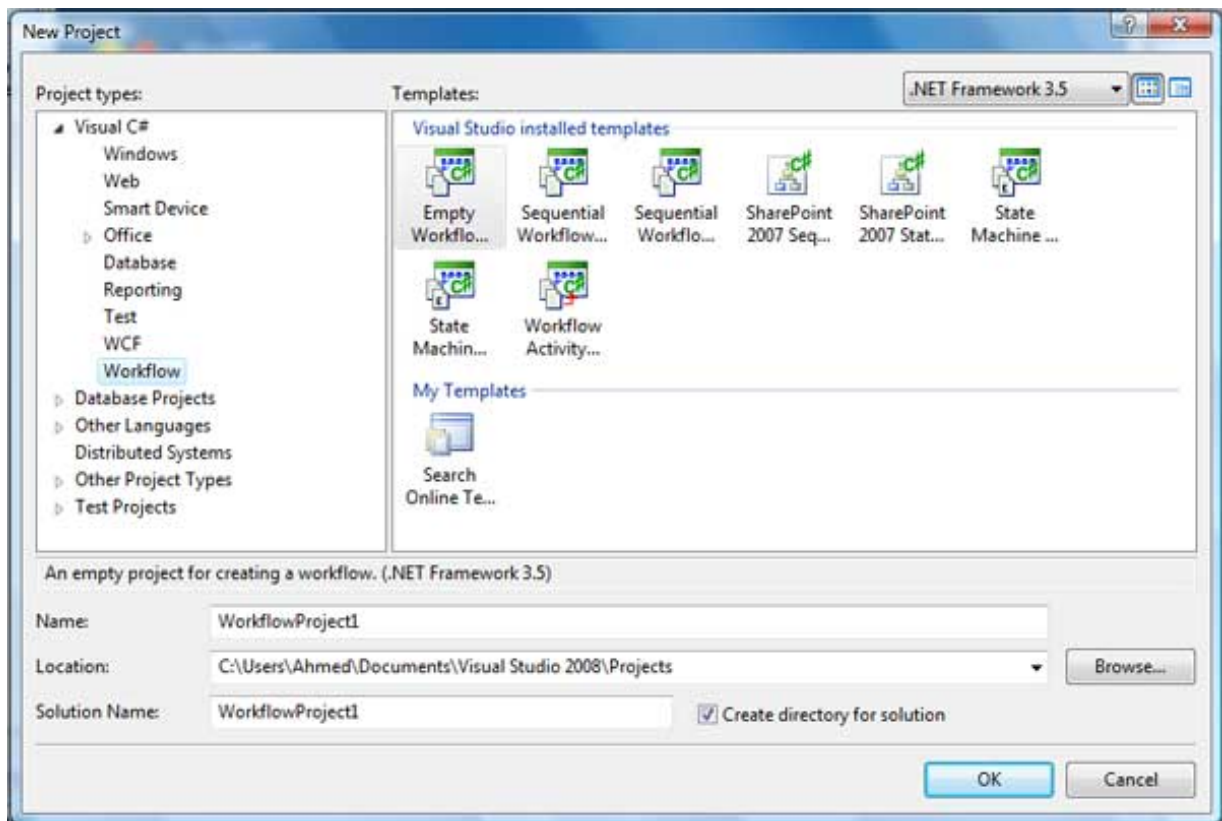
1. مكونات و اساسيات WF

ال WF هي كود قابل للتنفيذ وليس فقط مجرد رسومات توضيحية مثل visio ، لذا فالجمل الشرطية والتكرارات وخلافه الموجودة في ال WF يتم تنفيذها فعلياً. ولهذا السبب فال WF تحتوي على namespaces و assemblies وخلافه شأنها شأن اي مكونات للnet.

Workflow Activity

يتم توصيف العمل الخاص بنا على شكل Activities ويتيح لنا الفيجوال ستوديو وسائل للربط والانتقال بينهم وبين البعض مثل if else أو Delay لفترة أو حصول Invoke معين او يتم تنفيذها على شكل خطوات معينة:

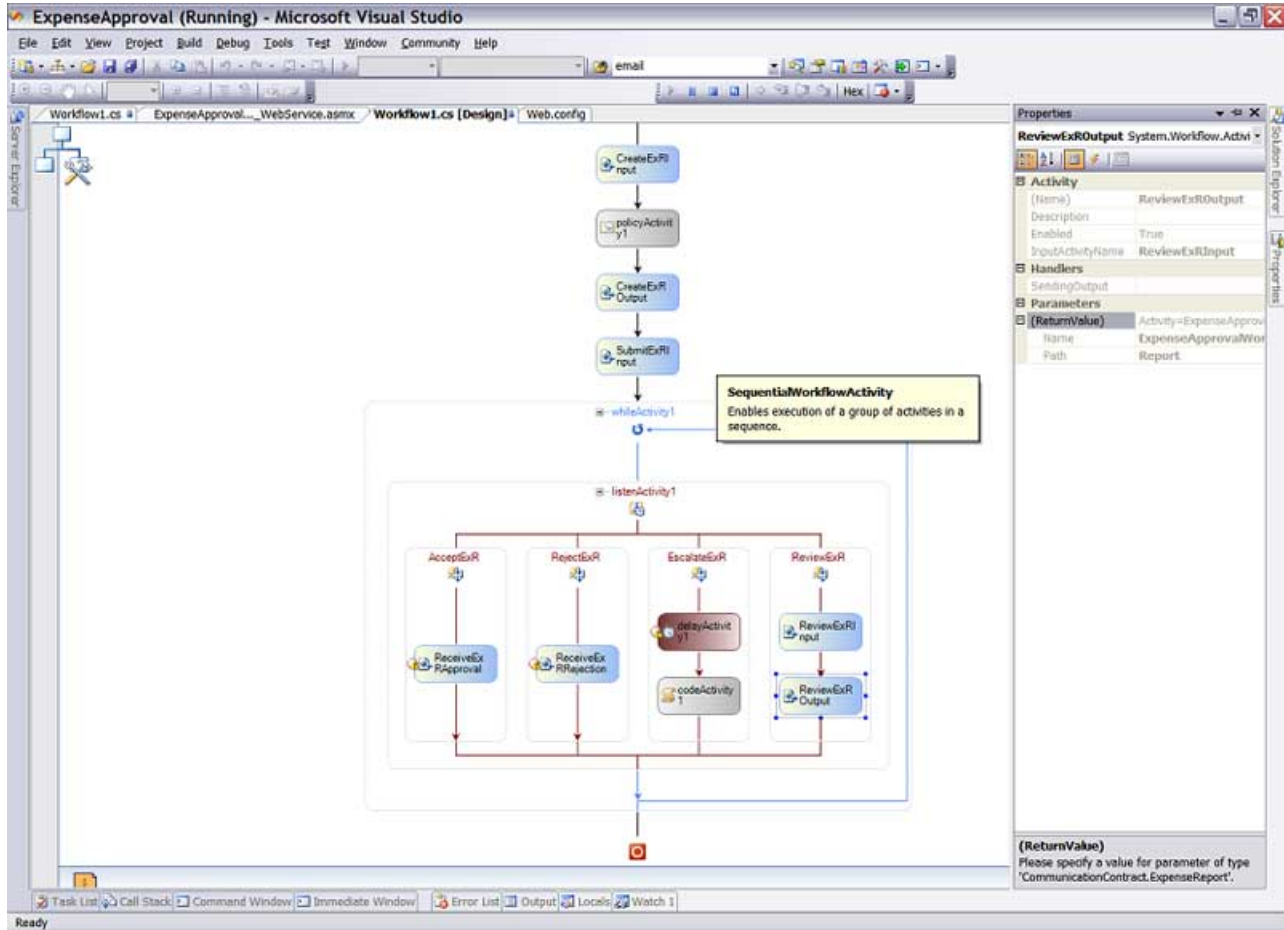
1. 1. البدأ مع WF

قم بانشاء مشروع جديد ، اجعل نوع المشروع Workflow ثم اختر Empty Workflow:

الصورة 19. 1. انشاء مشروع جديد من نوع Workflow.

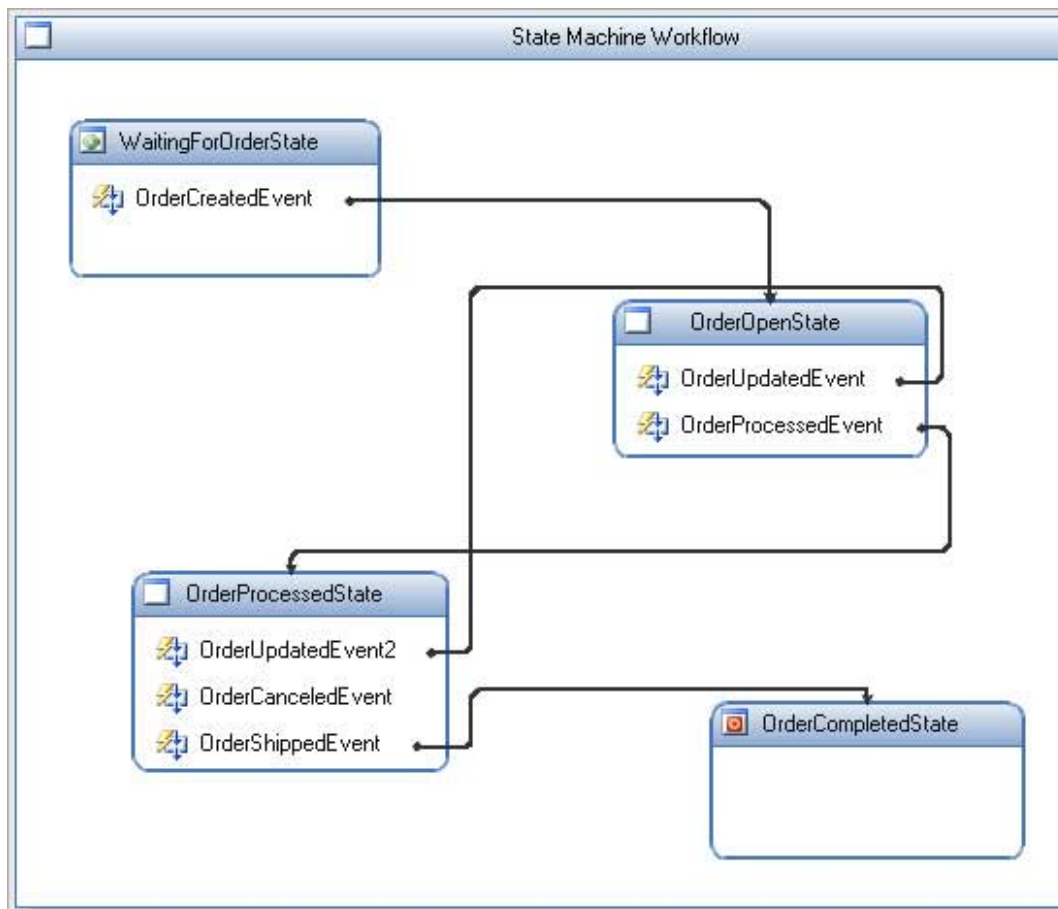
Empty Workflow

مشروع فارغ لا يحتوي على اي تنسيقات خاصة ، اضافة للأنواع التالية والتي تعد اساسيات عالم تصميم البرمجيات:

1. 2. Sequential Workflow

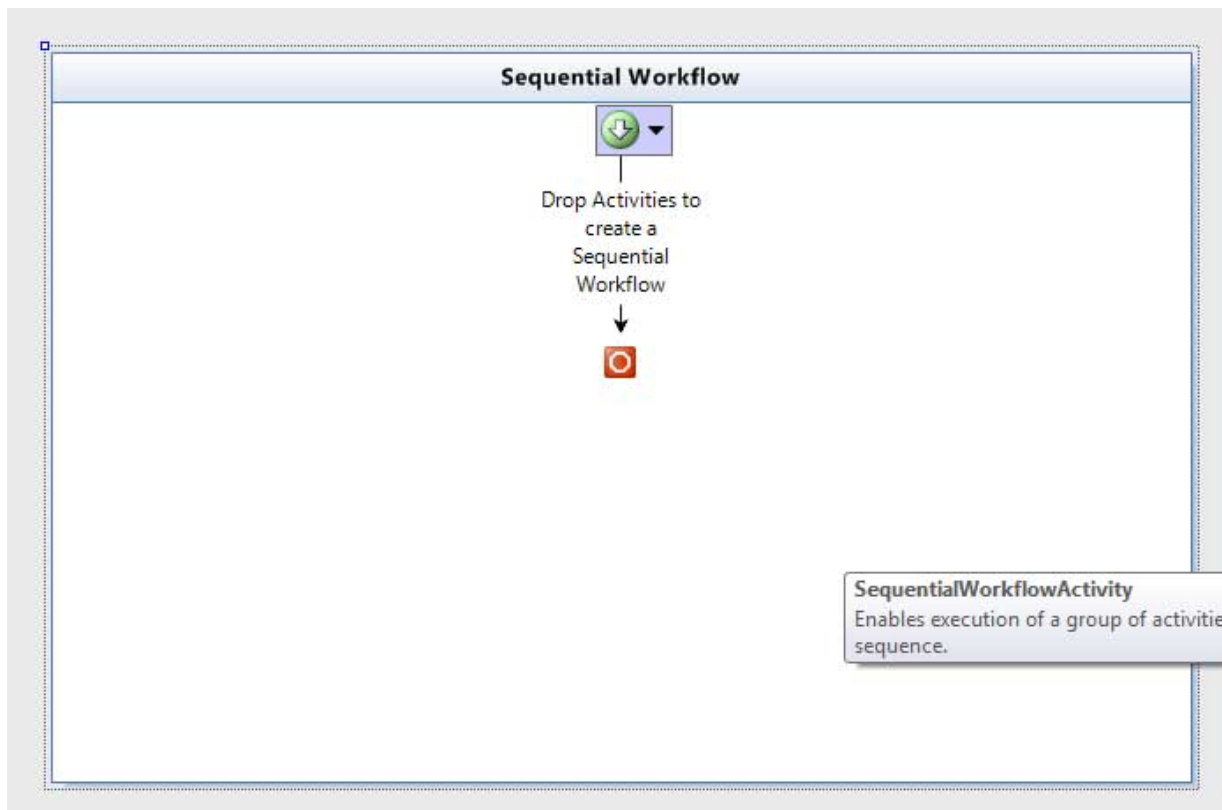
النوع الأول من انواع ال Diagrams التي توضح ال UML الخاص بنا ، يتم توضيح العمل فيه على شكل عمليات متلاحقة متتالية توضح الشكل العام لكامل العمليات ، وهو النوع الأشهر والأكثر استخداماً:

الصورة 19. 2. مخطط ال Sequential Workflows

1. 3. State Machine Workflow

تهتم اكثر ما تهتم بالأحداث عن طريق مجموعة من الطلبات والتغييرات:

الصورة 19. 3. مخطط ال StateMachine Workflow

2. تطبيق WWF

سنقوم الآن بعمل تطبيق بسيط ، سنقوم بالطلب من المستخدم ادخال كلمة المرور ، في حالة كون كلمة المرور صحيحة سوف نقوم بعرض رسالة ترحيب وإلا اعادة المحاولة لادخال كلمة المرور مرة أخرى.

لذا قم ببدء مشروع جديد Sequential Workflow Console Application وسمه كما تريد ، ستظهر لك شاشة المصمم بالشكل التالي كبداية:

الصورة 19. 4. مخطط ال Sequential Workflow

الآن سنقوم بادراج Code Activity وسنضع الأمر التالي في الحدث :codeActivity1_ExecuteCode

C#	كود

```
private void codeActivity1_ExecuteCode(object sender, EventArgs e)
{
    Console.ForegroundColor = ConsoleColor.Yellow;
    Console.WriteLine("***** First WPF Example *****\n\n");
    Console.ForegroundColor = ConsoleColor.White;
}
```

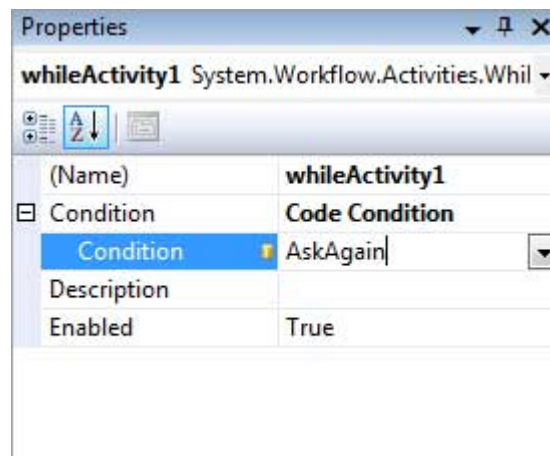
VB.NET	كود

```
Private Sub codeActivity1_ExecuteCode(ByVal sender As Object, ByVal e As
EventArgs)
    Console.ForegroundColor = ConsoleColor.Yellow

    Console.WriteLine("***** First WPF Example *****" & Chr(10) & "" & Chr(10)
& "")

    Console.ForegroundColor = ConsoleColor.White
End Sub
```

الخطوة التالية هي اضافة حلقة تكرارية لا يتم الخروج منها الا بادخال كلمة مرور صحيحة ، لذا قم بسحب While Loop ، وسنقوم بتحديد ال Condition الذي ستخرج منه اما بناء على:

- دالة function تعود بقيمة منطقية Boolean بحيث يتم استدعائها مع كل Loop وتعيد false للتوقف أو true للاستمرار.
- او بناء على declarative rule condition اي عن طريق جملة خاصة.

في مثالنا هذا سنعرف دالة AskAgain والتي ستعيد لنا true في حالة الجملة الخاطة لاعادة السؤال فيما تعيد لنا false في حالة الوصول لكلمة السر الصحيحة.

لذا من الشاشة الجانبية اكتب اسم الدالة في الشرط Condition بالشكل التالي - بعد تعريف خاصية بكلمة المرور اولاً: -

الصورة 19. 5. خصائص ال While Activity

وكود الدالة:

كود C#

```
public string Password { get; set; }
private void AskAgain(object sender, ConditionalEventArgs e)
{
    Console.Write("Enter Password: ");
    Password = Console.ReadLine();
    if (Password == "Ahmed Gamal")
        e.Result = false;
    else
        e.Result = true;
}
```

كود VB.NET

```
Public Property Password() As String
    Get
    End Get
    Set(ByVal value As String)
    End Set
End Property
Private Sub AskAgain(ByVal sender As Object, ByVal e As ConditionalEventArgs)
    Console.Write("Enter Password: ")
    Password = Console.ReadLine()
    If Password = "Ahmed Gamal" Then
        e.Result = False
    Else
        e.Result = True
    End If
End Sub
```

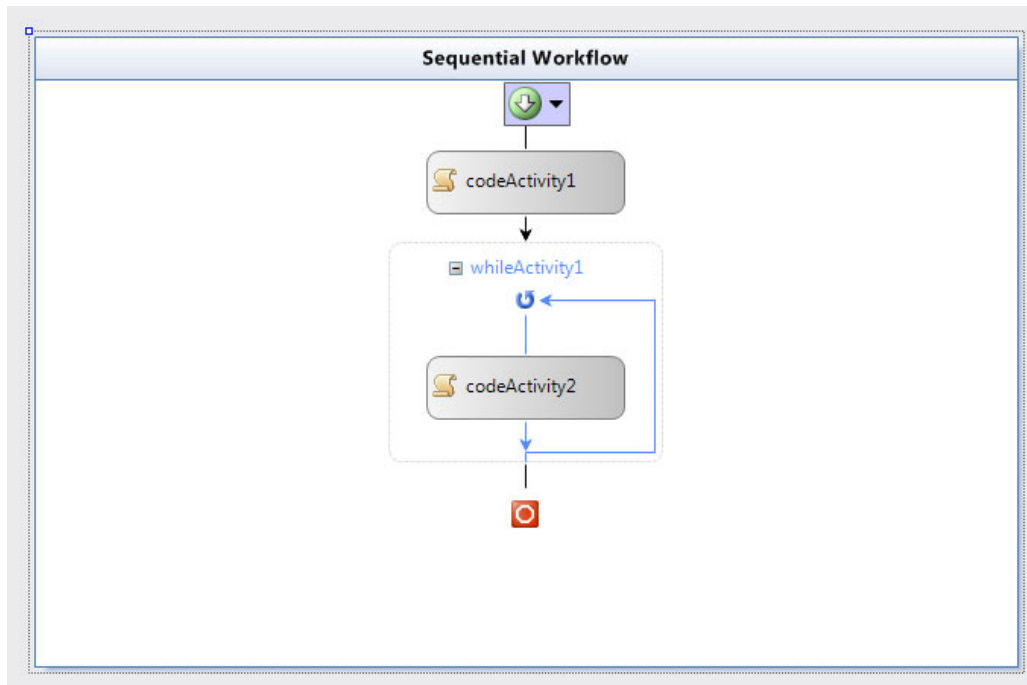
الآن خطوتنا الأخيرة هي اضافة Activity فيما لو لم يتم ادخال كلمة مرور صحيحة ، سيتم فيها طباعة كلمة This Password is Wrong ، لذا قم بسحب Activity وضع هذا الكود فيها:

كود C#

```
private void codeActivity2_ExecuteCode(object sender, EventArgs e)
{
    Console.WriteLine("This Password is Wrong...");
}
```

كود VB.NET

```
Private Sub codeActivity2_ExecuteCode(ByVal sender As Object, ByVal e As
EventArgs)
    Console.WriteLine("This Password is Wrong...")
End Sub
```

هكذا سيكون الشكل الكامل لل Sequential WF بالشكل التالي:

الصورة 19. 6. مخطط التتابع Sequential Workflow

و شاشة النتائج:

```
C:\Users\Ahmed\Documents\Visual Studio 2008\Projects\WorkflowConsoleApplication1\WorkflowC...
***** First WPF Example *****

Enter Password: sami
This Password is Wrong...
Enter Password: ahmed
This Password is Wrong...
Enter Password: gamal
This Password is Wrong...
Enter Password:
This Password is Wrong...
Enter Password:
This Password is Wrong...
Enter Password:
This Password is Wrong...
Enter Password:
This Password is Wrong...
Enter Password: Ahmed Gamal_
```

الصورة 19. 7. نتائج التنفيذ

*** لاحظ ان مثل هذا ال WF يمكن تطبيقه على نظام دخول معقد جداً كما هو الحال في نظام دخول بسيط ، نفس الهيكل مع اختلاف محتويات الكود لكل منهما ، لكن الصورة العامة واضحة وهو ما يفيد كثيراً في تسهيل قراءة وفهم محتويات الكود اضافة لعمليات ال trace وال Debug.

3. WF Code Library

في هذه النوعية اصبح بامكاننا انشاء WF يمكنه العمل مع ASP.net أو مع Windows Forms وليس فقط Console ، حيث سيتم تحويلها إلى ملف dll. ومن ثم استخدامها في اي تطبيق آخر.

الباب

20

WCF

من خلال استعراضنا للتقنيات الجديدة في عالم net. في مطلع الكتاب ذكرنا بإنها تقنية جديدة موجهة للتطبيقات التي تعمل على الشبكات أو لتلك التطبيقات الموزعة وخلافه ، وذكرنا ضمناً أنها استبدال للتقنيات الجديدة بدءاً من Winsock وانتهاء بـ Sockets .

وقبل البدء بعالم WCF لنلق نظرة سريعة على عالم Web Services منذ البداية.

1. Web Service

مبدئياً خدمات ويب أو ما يطلق عليها اسم Web Service هي عبارة عن برامج net. طبيعية أو على سبيل التحديد فهي ASP.net ولكنها ليس لها واجهة (المقصود هنا بالواجهة هي واجهة الاستخدام وليست Interface) ويستطيع المستخدمون الوصول لها بواسطة Interface معين ، حيث يتلق أمر أو طلب Request ثم يستجيب لها عن طريق http protocol اعتماداً على معاييرْ XML والتي تؤمن بالتالي أن يفهمها أو تفهمها ملايين البرامج والأجهزة ومواقع الإنترنت ببساطة تامة.

والآن سننسى التعريف السابق والمقدمة وسنحاول ان نفهم الموضوع بطريقة أخرى ، فخدمة ويب عبارة عن برنامج بدون واجهة يقوم المستخدم بمناداته وطلب بعض المعلومات منه ومن ثم ترد عليه الخدمة بهذه المعلومات ، وواحد من أشهر التطبيقات لذلك هي خدمات ويب الخاصة بالطقس ، حيث تطلب منها الطقس في مدينة معين لتعود لك بدرجة الحرارة مثلاً ، وكل ذلك أن يكون لها واجهة استخدام اضافة إلى أنها تكون موجودة على الإنترنت.

وكما أسلفنا فخدمات ويب تعتمد على XML كمعيار لتمثيل البيانات ، ما يعني أن جميع البرامج في جميع الأماكن قادرة على التعامل معها وفهمها بغض النظر عن نظام التشغيل أو لغة البرمجة.

وبما أن الخدمات برامج كما أسلفنا سابقاً ، فإن السيرفر الذي سيستضفيها على الإنترنت لا بد أن يدعم هذا النوع من البرمجة ، وإذا أردت أن تجرب خدمتك الخاصة على جهازك الشخصي فتأكد من أنك قم بتحميل برنامج IIS من ابتداء من مايكروسوفت Windows 2000 وأكثر.

1 .1. عمل WebService خاص بك من خلال net.

في هذا الدرس السريع سوف نقوم بعمل Web Service نقوم باعطائها درجة الحرارة من وحدة (درجة فهرنهايت) لتقوم بتحويلها إلى وحدة (درجة مئوية) ، والمعادلة العامة هي:

```
result = (input-32) * 5/9
```

وللبدء قم باختيار مشروع جديد New Project واختر ASP.net Web Service ، واختر اسماً لها في localhost لكي يتم تجربتها على حاسبك الشخصي ، مثلاً:

http://localhost/ConvertWebService

سيقوم فيجوال ستوديو بانشاء بعض الملفات منها ملف Web Config تماماً كما فيASP.net ، كما ستجد Global.asax أما الصفحات الرئيسية فسوف تنشأ باسم Service1.asmx.
لاحقاً سيكون الاسم Service1 هي الطريقة التي يتم بها الوصول إلى ال WebService الخاص بك ، لذا لا تنسى تغييرها إلى اسم مناسب لك وليكن ConvertWS مثلاً..
والآن تماماً كما تقوم ببرمجة أي تطبيق باستخدام net. قم بتطبيق المعادلة السابقة ، لكن لا تنس أن Web Service لا بد ألا تحتوي على أدوات مرئية TextBox مثلاً.

ولذا قم باضافة كود بالشكل التالي VB.net مثلاً:

كود C#

```
[WebMethod(Description="الحرارة درجات تحويل لعملية مثال")]
public decimal Convert(decimal degree)
{
    decimal result;
    result = (degree - 32) * 5 / 9;
    return result;
}
```

كود VB.NET

```
<WebMethod(Description:="الحرارة درجات تحويل لعملية مثال")> _
Function Convert(ByVal degree As Decimal) As Decimal
   Dim result As Double
   result = (degree - 32) * 5 / 9
   Return result
End Function
```

والآن لنجرب تطبيق هذه الخدمة، ولا تنس أن يكون خادم ال IIS موجوداً على جهازك.

قم بفتح المتصفح وليكن Internet Explorer وقم بكتابة السطر التالي:

http://localhost/ConvertWebService/ConvertSW.asmx

تستطيع الآن تجربة الخدمة الخاصة بك والتأكد من كونها تعمل بصورة صحيحة.

وفي الواقع فلن يتم استخدام الخدمة بهذا الشكل ، وإنما هذه الطريقة تستخدم فقط لتجربة الخدمة على جهاز المستخدم والتأكد من أنها تعمل بكفاءة.

يتم استخدام WebMethod في وضع عدد من الخصائص لخدمة الويب ، نذكر منها على سبيل العجالة:

1- BufferResponse : وهذه الخاصية تحدد فيما إذا كان سيتم عمل buffering قبل الارسال للمستخدم ، ولها قيمة True أو False .

2- CacheDuration : وهي تحدد الوقت الذي يتم من خلاله عمل Cash وتأخذ قيمة رقمية بالثانية وتحدد الوقت المستغرق قبل اعادة ارسال البيانات من جديد.

3- EnableSession : تحدد فيما إذا تم تفعيل خاصية Session أم لا.

1. 2. استخدام WebService خاص بك في مشروعاتك الفعلية

لنفترض أننا نريد الاستفادة من خدمة ويب في برنامج مثلاً للعمليات الحسابية ، ونريد أن نستخدم Web Service الذي قمنا به في الدرس السابق والذي يقوم بالتحويل من فهرنهايت إلى درجة مئوية والذي كان له الإسم ConvertSW .

قم الآن بانشاء تطبيق ويندوز عادي ، ومن قائمة Project اختر Add Web Refernce .

سيظهر لك مربع حواري قم باختيار ConvertSW.asmx الذي قمنا ببرمجته في الدرس السابق ، وكخدمة اضافية من Viusal Studio فسيتم عرض الخصائص والدوال التي تحتويها هذه الخدمة.

والآن ننتقل سريعاً إلى الكود :

كود	C#

```
LocalHost.ConvertSW conv = new LocalHost.ConvertSW();
```

كود	VB.NET

```
Dim conv As New LocalHost.ConvertSW()
```

والآن قم مثلاً بوضع Textbox1 و Label1 حيث أنك تقوم بادخال درجة الحرارة في الأول وتنتظر لكلي تعرض لك في الثاني ، الكود الكامل لهذه العملية يحتوي على الأمر السابق الخاص بتعريف الأوبجكت conv ، اضافة للسطر التالي:

كود	C#

```
Label1.Text = Convert.toString(conv.Convert(textBox1.text));
```

كود	VB.NET

```
Label1.Text = Convert.toString(conv.Convert(textBox1.text))
```

التنبيه الأخير قبل انهاء هذه الدرس هو انك إذا قمت بعمل تعديل في الخدمة ، قم في البرنامج الذي يستخدمها بعمل تحديث Update لها عن طريق الضغط عليها بزر الماوس الأيمن ومن ثم اختيار. Update Web Reference

1. 3. استخدام خدمات الانترنت الجاهزة

سوف نقوم الآن بعمل مشروع يستخدم أحد خدمات ال Web Service الموجودة على الإنترنت ، وبعد البحث عثرت على الموقع التالي الذي يقدم بعض الخدمات:

رابط

http://www.webservicex.net/WCF/

يحتوي هذا الموقع على حوالي 70 خدمة مجانية ، قمت باختيار واحدة منها وهي

رابط

http://www.webservicex.net/ValidateEmail.asmx

وتخبرنا هذه الخدمة فيما إذا كان هناك ايميل بهذا الشكل أم لا وتبحث عن هذا الايميل في الشركات التي تقدم خدمات البريد الإلكتروني، وذلك عن طريق دالة اسمها : IsValidEmail وتعود بقيمة True او False.

والآن كما اعتدنا ، نقوم باضافة Web Refernce جديد للمشروع بالمسار الذي قمنا باضافته سابقاً ، ومن ثم سنقوم برسم على الفورم بحيث يمكن للمستخدم التعامل مع هذه الخدمة بسهولة، ولذلك سوف نقوم برسم مربع نص يتم فيه ادخال البريد الإلكتروني ، ومن ثم زر أمر بحيث يتم اختبار قيمة الشرط ومن ثم عرض رسالة MessageBox فيها حالة الايميل.

الكود الذي سيتم وضعه في زر الأمر هو:

كود C#

```
net.webservicex.www.ValidateEmail x = new
WindowsApplication1.net.webservicex.www.ValidateEmail();
bool r = x.IsValidEmail(textBox1.Text);
if (r)
    MessageBox.Show("ايميل صحيح ");
else
    MessageBox.Show("ايميل خاطئ ");
```

كود VB.NET

```
Dim x As net.webservicex.www.ValidateEmail = New
WindowsApplication1.net.webservicex.www.ValidateEmail()
Dim r As Boolean = x.IsValidEmail(textBox1.Text)
If r Then
    MessageBox.Show("ايميل صحيح ")
Else
    MessageBox.Show("ايميل خاطئ ")
End If
```

مع استمراري في البحث في هذا الموقع وجدت خدمة أخرى جديرة بالملاحظة وشائعة الاستخدام، وهي الخاصة بالتعرف على الطقس والمناخ وخلافه، تجد الخدمة هنا:

رابط

http://www.webservicex.net/globalweather.asmx

المثال الأول لاستخدام هذه الخدمة هي معرفة المدن الموجودة في دولة معينة باستخدام GetCitiesByCountry، مثلاً هذا الكود:

كود C#

```
net.webservicex.www1.GlobalWeather m = new
WindowsApplication1.net.webservicex.www1.GlobalWeather();
MessageBox.Show(m.GetCitiesByCountry("Egypt"));
```

كود VB.NET

```
MessageBox.Show("Welcome")
Dim m As Net.webservicex.www1.GlobalWeather = New
WindowsApplication1.net.webservicex.www1.GlobalWeather()
MessageBox.Show(m.GetCitiesByCountry("Egypt"))
```

يعرض لنا أسماء المدن الكبرى في مصر، جرب الآن عمل TextBox وجرب الدول التي تريد معرفة المدن الكبرى لها...

الدالة الأخرى هي معرفة حالة الطقس باستخدام GetWeather ، بالشكل التالي مثلاً:

كود C#

```
net.webservicex.www1.GlobalWeather m = new
WindowsApplication1.net.webservicex.www1.GlobalWeather();
MessageBox.Show(m.GetWeather("Asyut", "Egypt"));
```

كود VB.NET

```
Dim m As net.webservicex.www1.GlobalWeather = New
WindowsApplication1.net.webservicex.www1.GlobalWeather()
MessageBox.Show(m.GetWeather("Asyut", "Egypt"))
```

والآن بقي أن أشير إلى أن اشهر الخدمات الموجودة على النت هي تلك المتعلقة بالتحويلات المالية ، حيث تحدث قيم هذه الخدمات فورياً بحيث يمكنك استخدامها في برنامجك لتحويل العملات بدلاً من الاعتماد على المعادلات الحالية والتي تتغير باستمرار.

هناك أيضاً خدمات أكثر تعقيداً مثل الاتصال بالبنوك والدفع عن طريق البرنامج باستخدام خدمة يقدمها البنك وخلافه . وهي قصة طويلة لكنها تسير في اتجاه مشابه.

أيضاً هناك خدمة RSS وهي تسير في اتجاه مشابه ، خدمة أخرى منتشرة وهي خدمة التعرف على حالة الطقس ، المناخ ، وخصائص الدول والعلم والعملة والسكان وخلافه ، نوع آخر وهو التعرف

على معلومات ISPN من الشركة حيث قد يفيدك في بعض البرامج ، أيضاً ستجد خدمات ويب لكل ما يخطر ببالك من التحويلات والقياسات المعروفة.

2. مقدمة إلى WCF

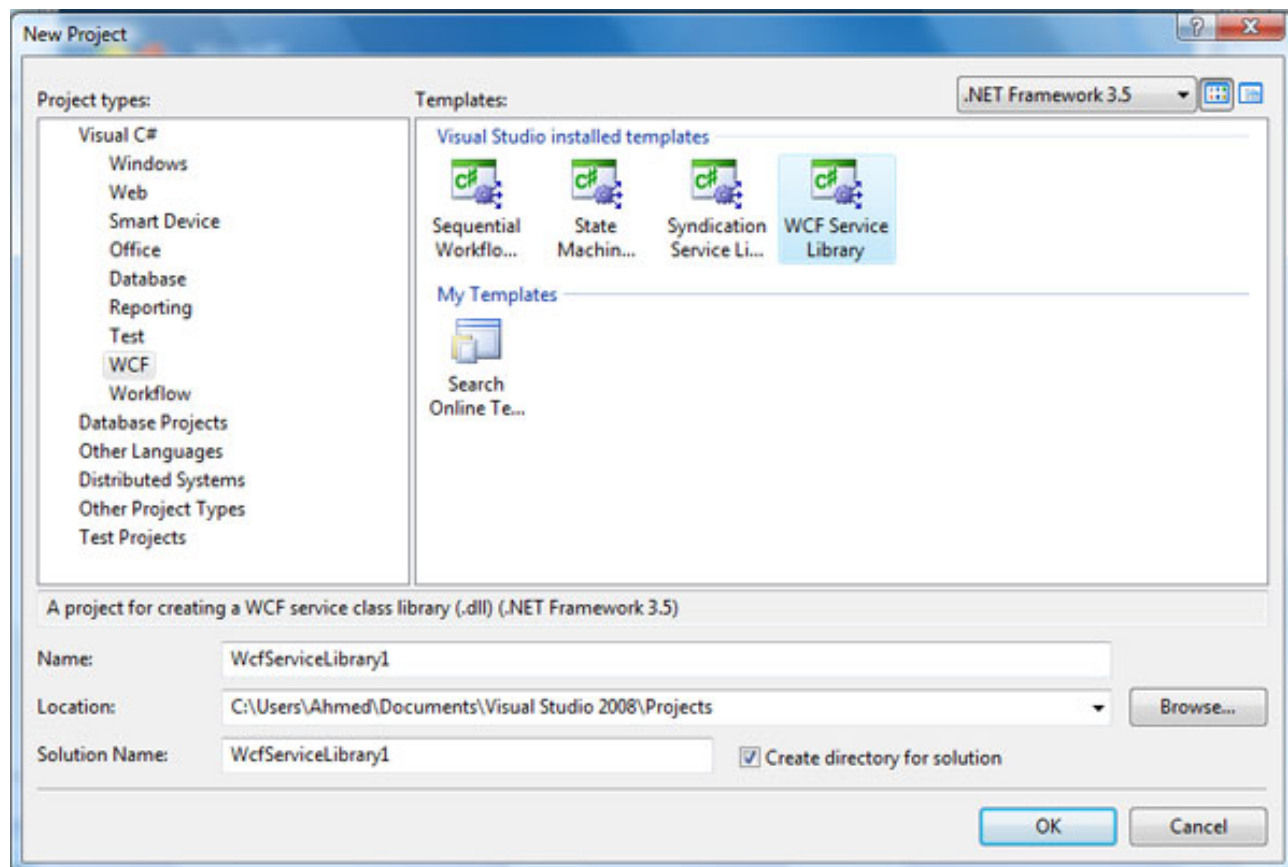
واحدة من التقنيات الجديدة التي واكبت ظهور NET 3.0. كانت تقنية Windows Communication Foundation والتي يرمز لها اختصاراً WCF ، وهي مجموعة من الدوال API's التي تختص بعالم الاتصال وال web services وال remote access وخلافه.

النقطة التي نتحدث عنها وستكون شغلنا الشاغل هنا هو كيفية التخاطب بين برنامجين على جهازين x و y حتى في حالة x=y بحيث تسمح لنا هذه العملية بعمل التطبيقات على الشبكات والوصول للأجهزة عن بعد وبرامج المحادثة ونقل الملفات ... الخ من التطبيقات التي تميز عصر الإنترنت الحالي.

هناك الكثير من التقنيات التي ظهرت لدعمك كمبرمج في تطوير برامج من هذه النوعية ، حيث وفرت لك مجموعة من الحلول لتسهيل التعامل مع ال API's الخاصة بعمليات الشبكات وخلافه، مثل DCOM ، MSMQ ، ومع ظهور تقنية net. تم تقديم مجال الأسماء System.Runtime.Remoting الخاصة بمثل هذه العمليات ، هناك ايضاً تقنية Web XML Services، تمكنك هذه التقنيات من تسهيل عمليات عمل نظم موزعة بصورة كبيرة جداً ، وإن كنت لا ترغب في العمل في هذا المستوى العام فيمكنك التخصيص زيادة والعثور على خيارات اكبر واوسع مقابل زيادة بسيطة في نسبة التعقيد عن طريق استخدام Named Pipes, Sockets, and P2P وهي الاشهر فعلياً بين مستخدمين ال net. على الأقل في وطننا العربي نظراً لعدم شهرة التقنيات السابقة - برغم انها توفر الكثير من الوقت فعلياً ، يمكنك البدء من الدرس السابق كمثال.

كانت هذه الحياة لك كمبرمج net. قبل عالم WCF ، في الدرس القادم سنبدأ بالتعرف على طبيعة هذه التقنية.

2 .1. البداية مع ال WCF

بعد تشغيلك لـ Visual Studio 2008، قم باختيار نوع المشروع WCF ثم WCF Service Library بالشكل التالي مثلاً:

الصورة 20 .1. اضافة مشروع جديد WCF.

هناك خيارات أخرى تقدمها لك WCF بحيث يمكنك تطبيقها مع خدمة RSS أو تقنية WWF افتراضياً ، هناك ايضاً ضمن اختيارات انشاء Web Site جديد تجد WCF Service ولكن ليس هذا هو موضوعنا الآن.

الآن تجد نفسك قد قمت افتراضياً بانشاء ثلاث ملفات assembly هي:

- WCF Service
- WCF Service host

- WCF client

وللربط لا بد من توافر المعلومات الثلاث التالية:

- Address: هو يحدد العنوان الذي يتم الاتصال به.

- Binding:طرق الربط.

- Contract: صف المهام التي يتم تنفيذها من خلال WCF.

2 .2. شكل و محتويات ال Address

النقطة الأولى من معلومات WCF هي العنوان الذي ستتصل به ، له الصيغة العامة التالية:

رابط

```
scheme://<MachineName>[:Port]/Path
```

حيث أن scheme تحدد البروتوكول الذي سوف تعمل من خلاله ما بين البروتوكولات الموجودة

MachineName تحدد المكان الذي ستتصل به

Port هو المنفذ الذي سيتم الاتصال من خلاله ،

وأخيراً Path وهو مسار ال Services في الجهاز الذي تود الاتصال به.

هذا على سبيل المثال:

رابط

```
net.tcp://localhost:8080/MyWCFService
```

تجربة بسيطة:

قم بانشاء WCF Liabrary جديدة ، بعد انشاءها قم باضافة تعريف للدالة في IService1.cs:

كود C#

```
[OperationContract]
int Add(int x, int y);
```

كود VB.NET

```
<OperationContract()> _
Private Function Add(ByVal x As Integer, ByVal y As Integer) As Integer
End Function
```

ومن ثم في Service1.cs قم بكتابة الدالة التالية:

كود	C#

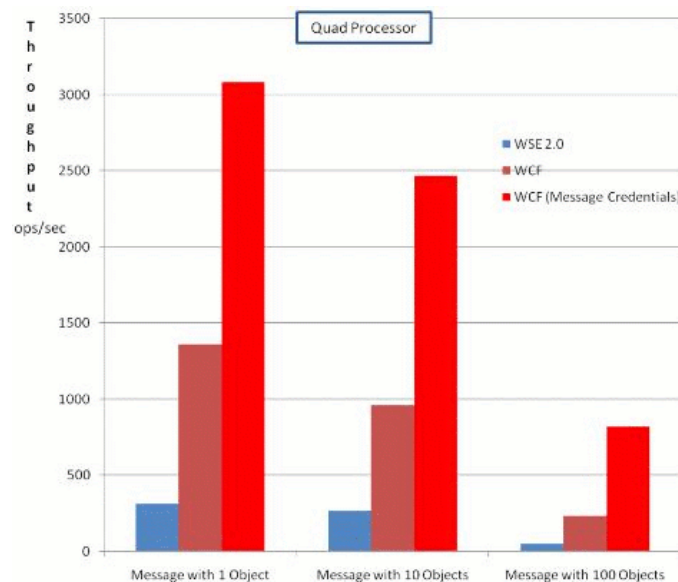
```
public int Add(int x, int y)
{
    return x + y;
}
```

كود	VB.NET

```
Public Function Add(ByVal x As Integer, ByVal y As Integer) As Integer
    Return x + y
End Function
```

وفقط ، قم بعمل Run وقم بتجربتها من خلال WCF Test Client بالشكل التالي مثلاً ، بعد كتابة رقمين قم بالضغط على Invoke لتجربة الحل.

أليس هذا أسهل بكثير من الدرس السابق حول Web Services ؟

ليس هذا فقط، بل سأنقل لك هذه المقارنة من موقع مايكروسوفت بين هاتين التقنيتين:

الصورة 20. 2. سرعة الأداء باستخدام تقنية ال WCF.

إذا كنت تود استعراض كافة أوجه المقارنة يمكنك الإطلاع عليها عبر الرابط التالي :

رابط

http://msdn.microsoft.com/en-us/library/bb310550.aspx

الباب

21

نقاط متقدمة

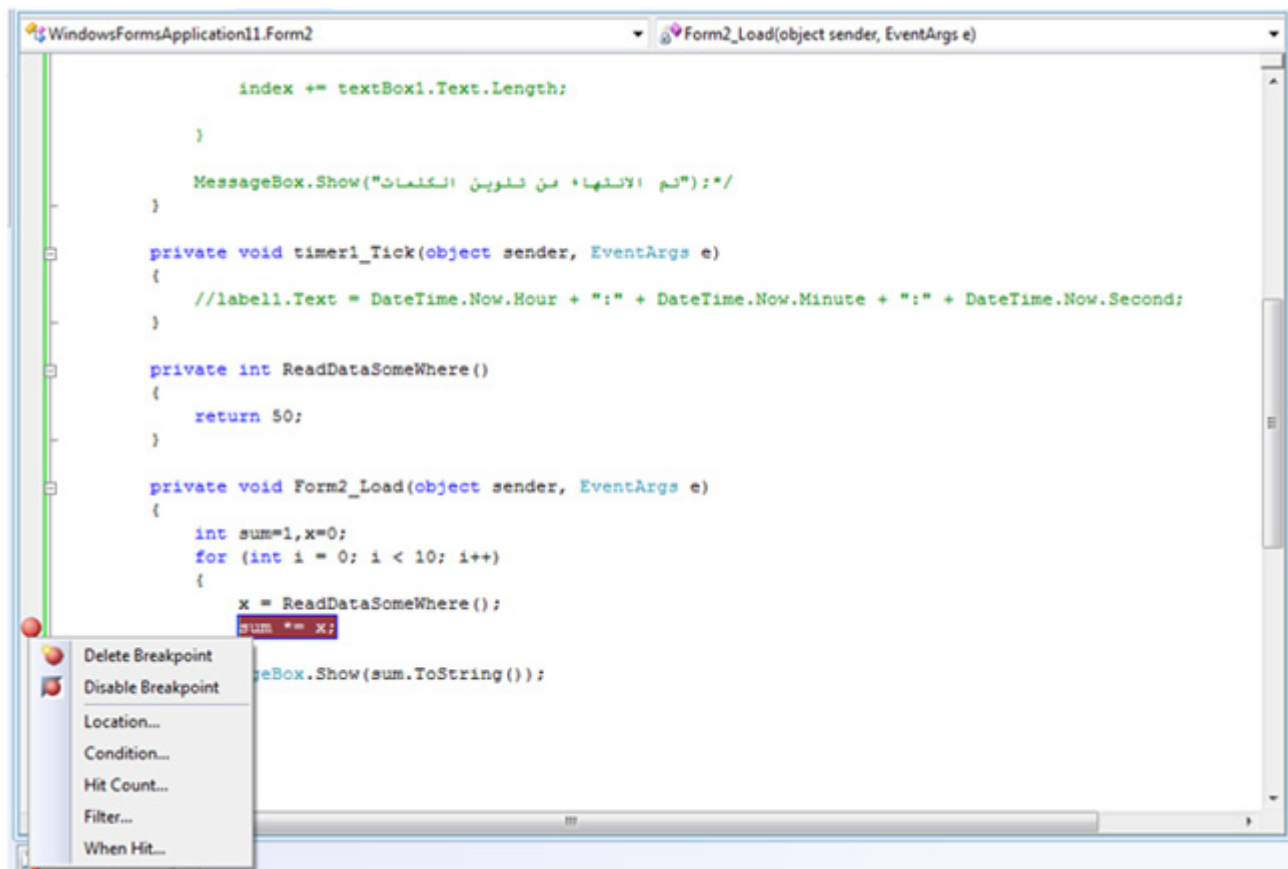
نستعرض هنا سوية بعض النقاط الإضافية التي لم نستعرضها خلال سير الدروس قبل الإنتقال إلى عالم تطوير المواقع .

1. التنقيح - Debugging

أثناء عملك على البرنامج تظهر لك أحياناً عدة أخطاء تشيب رأسك ، حيث لا تكون قادراً على معرفة المكان الذي ظهر منه الخطأ تحديداً ، أو أنك تجد فجأة قيمة متغير مختلفة عن القيمة التي كنت تتوقعها ، من هنا ظهرت فائدة ال Debug لتتبع أخطائك بصورة أفضل .

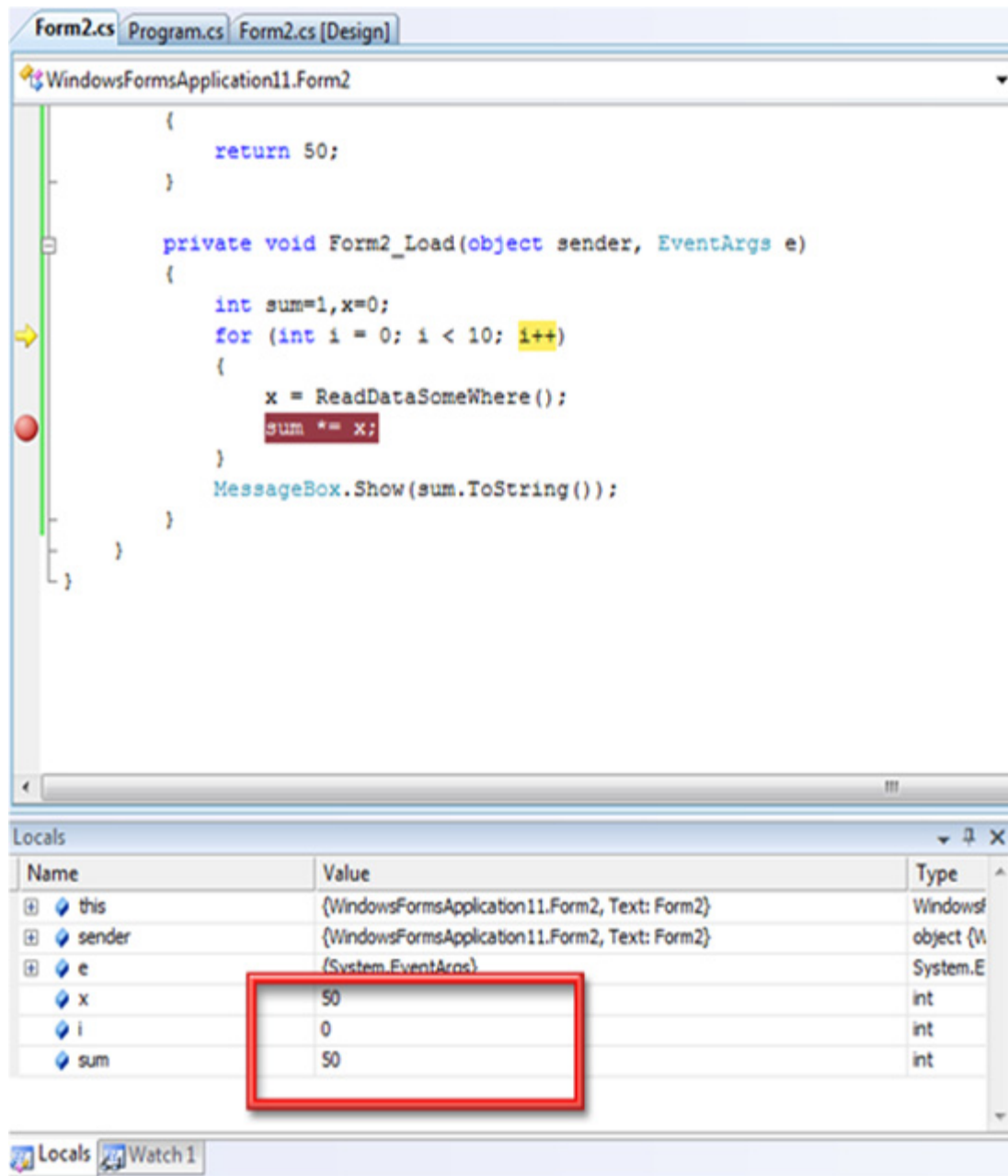
BreakPoints

نقاط يتم وضعها على أماكن معينة من الكود بحيث يتوقف التنفيذ فور الوصول إليها ويعيدك إلى ال Visual Studio، حيث يمكنك مراقبة قيم المتغيرات واستعراضها والتأكد من خلوها من المشاكل ، يمكنك اعادة استئناف التنفيذ سطر بسطر بالضغط على F11 أو مواصلة التنفيذ للنهاية بالضغط على F5 .

يمكنك اضافة BreakPoint بالضغط على F9 في السطر المحدد ، أو الضغط على جانب الكود بالشكل التالي مثلاً :

الصورة 21. 1. اضافة نقطة توقف BreakPoint.

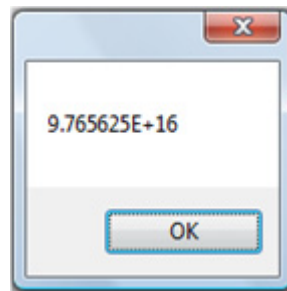
والآن ، كما ترى في مثالنا السابق نقوم بقراءة عدة بيانات ، إلا أني افاجأ دوماً بكون الناتج سالب برغم ان العملية ضرب رغم ان توقعي ان كل المدخلات لابد أن تكون موجبة وبقيم كبيرة، يا ترى ما هي المشكلة ؟؟؟

من اجل هذا وضعت هذه النقطة للمراقبة، وقمت بالبدء بتشغيل البرنامج ومن ثم التنقل بين الخطوة والأخرى F11.

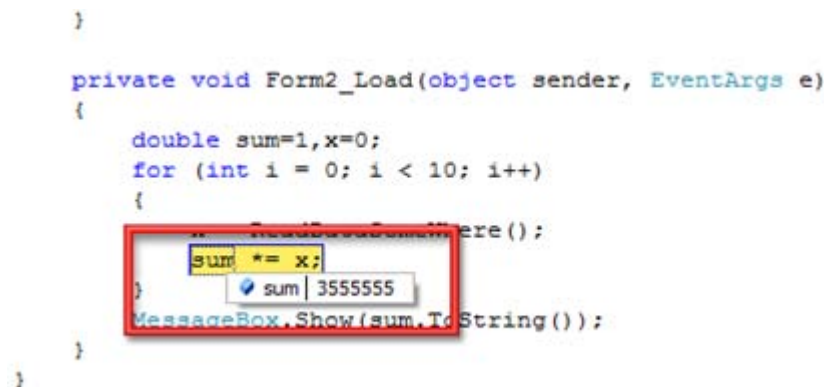
ستجد ان البرنامج سيقوم مباشرة بالتوقف وقت الوصول إلى هذا السطر ، وسيمكنك رؤية المتغيرات كلها في هذه الأثناء بالشكل التالي :

الصورة 21 .2. متابعة قيم المتغيرات المحلية.

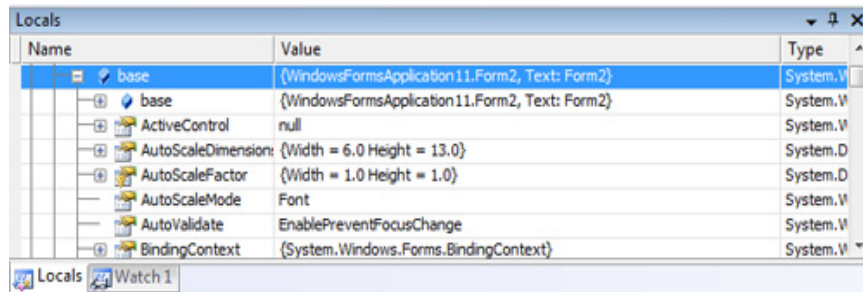
الآن استمر في الخطوات ، بعد 6 خطوات ترى ماذا حدث ؟

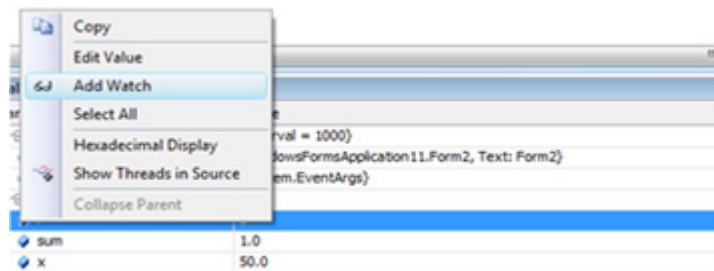
آه الآن ابتدأت في تخيل سبب المشكلة ، فعلاً عندما يصل الرقم لحد ما يبدأ الرقم يتحول إلى سالب ، المتغير لم يعد يستوعب رقماً اكبر من هذا .

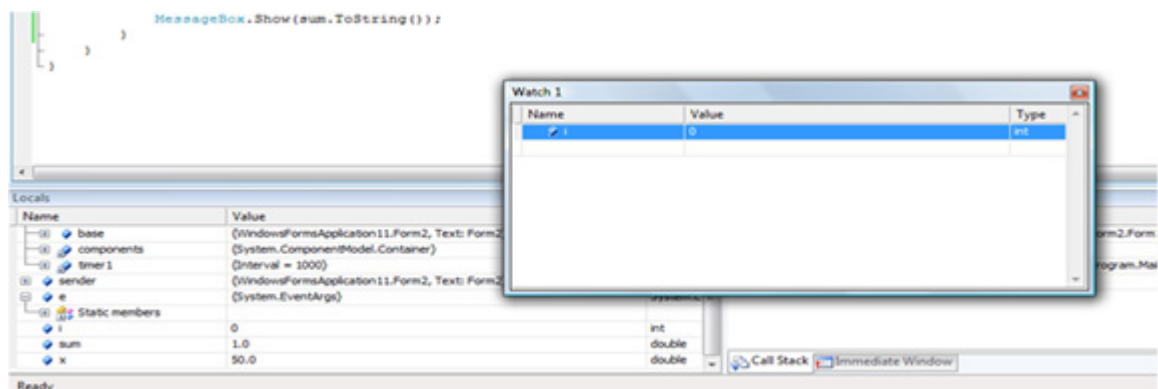
إذن الحل بتعريف ناتج عملية الضرب Double ... انتهت العملية وأمكنك الحصول على الناتج المطلوب :

هذه هي الفائدة التي نجنيها من عملية ال Debug ، في مشاريعك الكبيرة سيكون الموضوع أكثر صعوبة وتعقيداً وربما تحتاج لمراقبة قيم المتغيرات بين عدة دوال ، لذا سيكون الحل الأمثل لك بالاعتماد على هذه الأدوات .

ليس هذا فقط ، بل في هذه المرحلة يمكن تغيير قيم المتغيرات ، لهدف ما وجدت انك تحتاج إليه ستجد نافذة بالتغيير بالشكل التالي مثلاً :

```
}
private void Form2_Load(object sender, EventArgs e)
{
    double sum=1,x=0;
    for (int i = 0; i < 10; i++)
    {
        ...ere();
        sum *= x;     sum | 3555555
    }
    MessageBox.Show(sum.ToString());
}
}
```

الصورة 21. 3. تغيير قيم المتغيرات أثناء عملية التنقيح.

أيضاً كل المتغيرات والفئات الموجودة مراقبة بالشكل التالي :

الصورة 21. 4. مراقبة المتغيرات المحلية.

ولو رغبت في مراقبة متغير بعينه فيمكنك اختيار اضافة مراقبة خاصة Add Watch بالشكل التالي :

الصورة 21. 5. مراقبة المتغيرات المحلية.

حيث سيقوم بفتح نافذة مراقبة خاصة به وحده بالشكل التالي :

الصورة 21. 6. نافذة ال Watch.

2. تجهيز البرامج للتوزيع

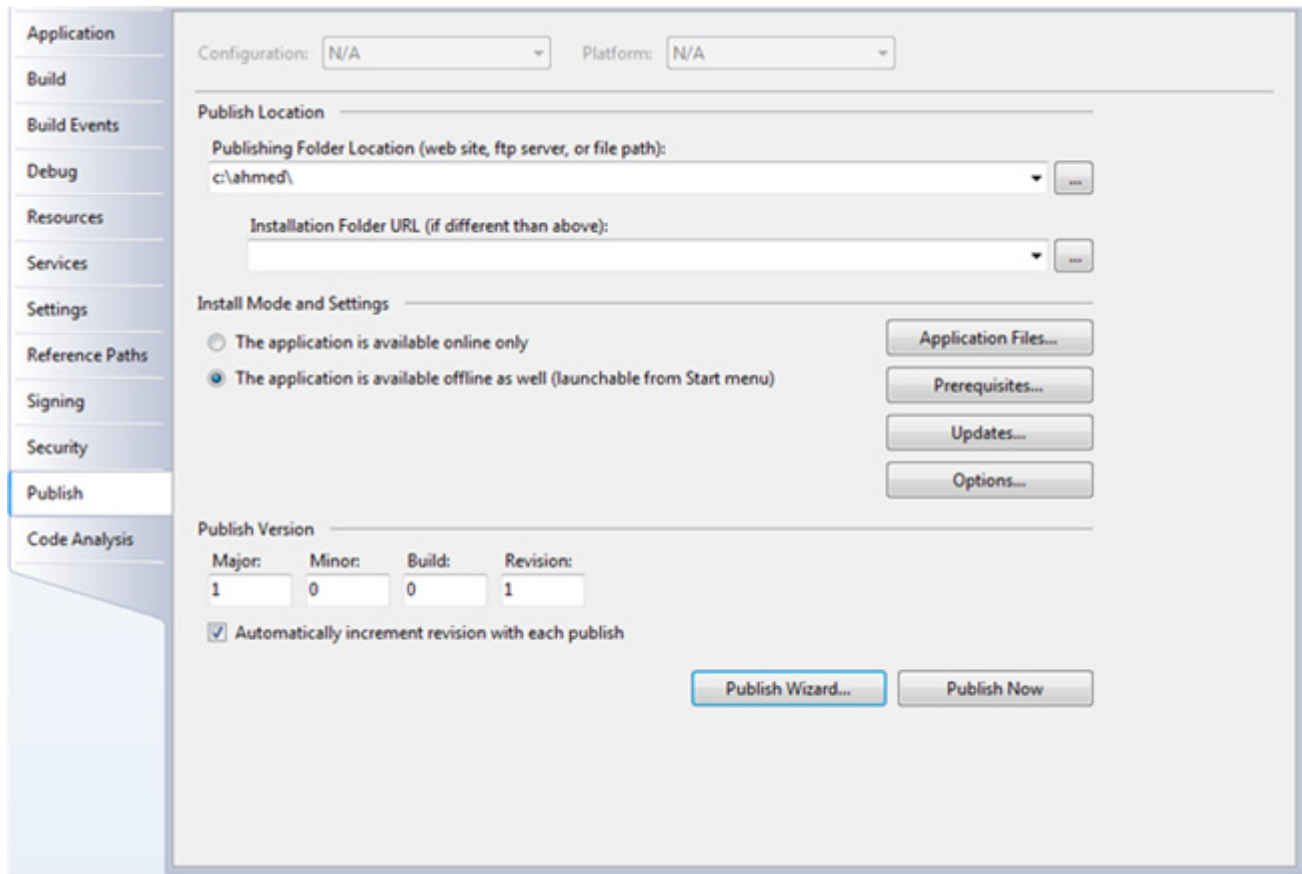
بعد ان نكون قد انتهينا من برامجنا ، نحن الآن بحاجة لنقلها إلى المستخدم ، ولما كان جهاز المستخدم قد لا يحتوي على كافة المكونات اللازمة لنا للعمل نقوم الآن بعمل Publish للبرنامج على شكل Setup ليقوم المستخدم بتشغيله مباشرة ، يتم ذلك من القائمة Build-Publish .

- بداية قم بتحديد المكان الذي تود النشر إليه ، ليكن C:\ahmed مثلاً .

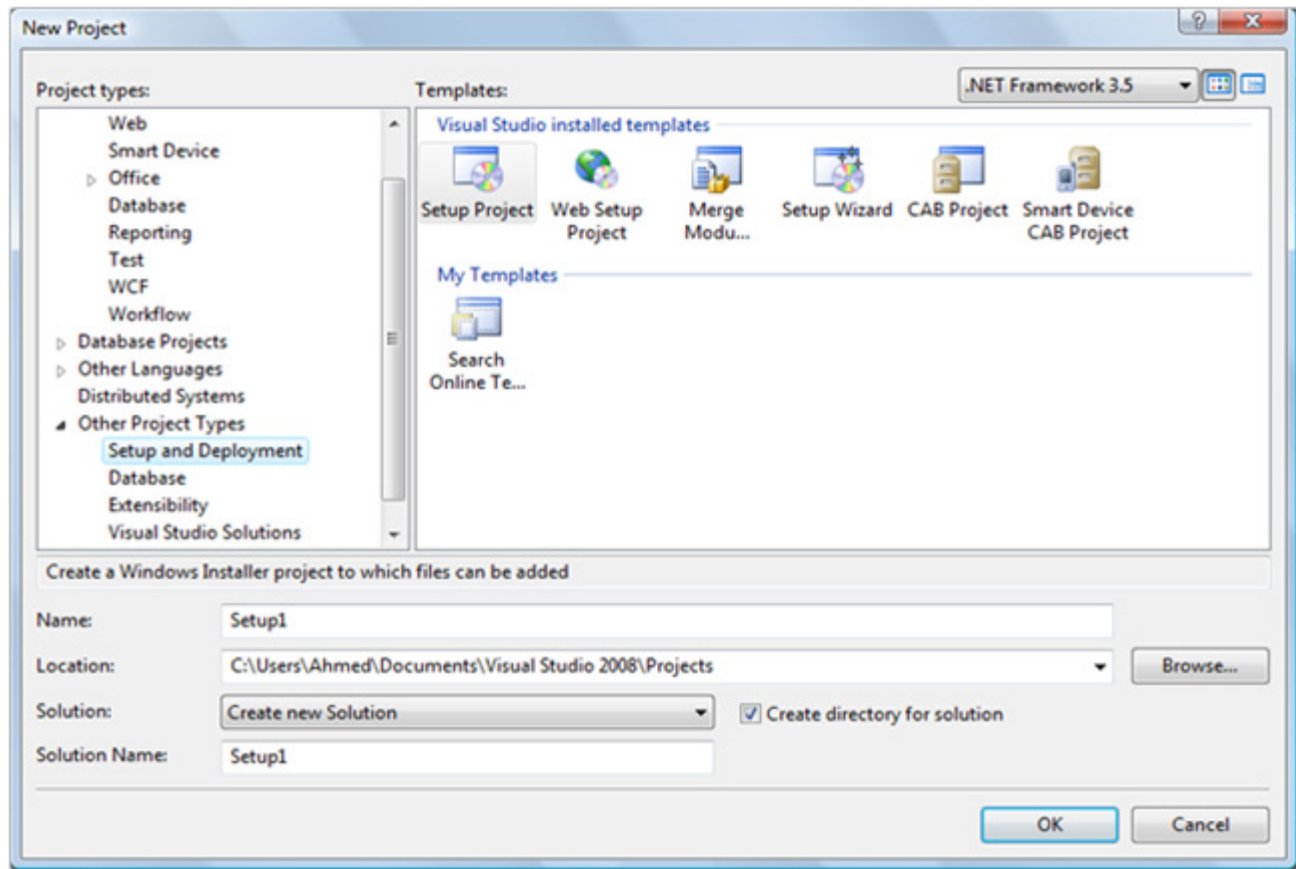
- في الخطوة التالية قم بتحديد المكان الذي تود للمستخدمين تحميل نسخة البرنامج منه ، سنختار CD-DVD .

- إذا كنت ترغب في أن يبحث البرنامج عن التحديثات في مكان معين فقم بذلك وإلا تجاهل هذه الخطوة .

- اضغط Finish .

- ستجد كل شيء تم تجهيزه ، يمكنك عمل بعض التغييرات عن طريق خيارات المشروع ثم Publish ، مثلاً لتضمين ملفات اضافية أو خلافه .

الصورة 21. 7. خصائص المشروع/ تبويب النشر.

- الآن اصبح برنامجك جاهزاً للعمل ، قم بالضغط على Setup وسوف يتم تحميل البرنامج في Program Files ويمكنك التعامل معه كبرنامج عادي جداً .

- يمكنك استخدام أدوات أكثر تقدماً من أجل هذه الموضوع مثل installshield و setup factory، أيضاً يمكنك الاستفادة من الخصائص المتقدمة التي توفرها لك مشروعات ال Setup في الفيجوال ستوديو من القائمة New Project حيث ستكون قادراً على التحكم في سير وشاشات عملية التحميل والاتفاقيات وخلافه:

الصورة 21. 8. اضافة مشروع تسطيب.

3. C# vs VB.net

لو لاحظت طوال فترة دروسنا اعتمادنا بشكل متوازي على لغتي C# و Vb.net ، فما هي الفوارق أو أوجه الشبه بينهما . لنعد أولاً إلى نقطة البداية لانطلاق ال net..

في بداية ظهور مبادئ ال net. كانت C# هي اللغة المعتمدة ، ولما كانت مايكروسوفت تمتلك لغة برمجة ذات سوق واسع في ذلك الوقت وهي فيجوال بيسك ، رأت مايكروسوفت ألا تخسر مبرمجيها بتطوير لغة موازية شبيهه في ال Syntax مع الفيجوال بيسك ولكنها تطبق مبادئ ال net. ، من هنا جاءت VB.net والتي تعتبر نسخة أخرى من ال C# ولكن بأسلوب ال VB ، وهذا هو الشبه الوحيد بينها وبين اللغة القديمة ، اسلوب الكتابة فقط ليس إلا .

ولذا إن حاولنا التحدث عن الشبه بين C# و VB.net فلن نستطيع استيفاءه ، الأسهل هنا هو سرد الاختلافات فهما في النهاية لغة واحدة ، سنبدأ بسرد الفروقات في أسلوب الكتابة ...

- أولاً : كونك مبرمج سي شارب لا تنس أن تضع ; في آخر كل سطر
- ثانياً : طريقة تعريف المتغيرات :

كود C#
```
int x;
ClassName x = new ClassName();
```

كود VB.NET
```
Dim X As Integer
Dim X As ClassName = New ClassName()
```

- ثالثاً : في حالة الشروط في VB.net اكتب = أما في C# اكتب == ، كما ستجد العلامة != لعدم المساواة بدلاً من <>.

- رابعاً : تعتمد لغة VB.net على جمل مثل End وتستخدمها في الاجراءات والفئات وغيرها مثلاً :

كود VB.NET
```
if x=1 Then
End if
```

بينما تجدها في C# باستخدام الأقواس :

كود C#
```
if (x == 1)
{
}
```

وكذلك في الفئات والاجراءات أيضاً :

كود C#
```
using System;
namespace MyNameSpace
{
    class HelloWorld
    {
        static void Main(string[] args)
        {
            System.Console.WriteLine("HelloWorld");
        }
    }
}
```

كود	VB.NET

```
Imports System
Namespace MyNameSpace
    Class HelloWorld
        'Entry point which delegates to C-style main
Private Function
        Public Overloads Shared Sub Main()
            Main(System.Environment.GetCommandLineArgs())
        End Sub
        Overloads Shared Sub Main(ByVal args() As String)
            System.Console.WriteLine("Hello World")
        End Sub 'Main
    End Class 'HelloWorld
End Namespace 'MyNameSpace
```

- خامساً ، في فيجوال بيسك .net سوف تستخدم And,Or,Not,OrElse بينما في C# سوف تستخدم && || ! .

- سادساً : عند جمع النصوص استخدم & في VB.net ولكن في C# استخدم + .

- سابعاً : else if في C# هي ElseIf في VB.net

- ثامناً : كما اسلفنا في موضوع End ، ففي الحلقات التكرارية تعتمد C# على الأقواس أيضاً :

كود	C#

```
for (int i = 2; i <= 10; i += 2)
{
    System.Console.WriteLine(i);
    System.Console.WriteLine(i*10);
}
```

وفي حالة كونها سطر واحد يمكن الاستغناء عن الأقواس :

كود	C#

```
for (int i = 2; i <= 10; i += 2)
    System.Console.WriteLine(i);
```

أما في VB.net فهي تأخذ شكلاً موحداً :

كود	VB.NET

```
For c = 2 To 10 Step 2
    System.Console.WriteLine(c)
Next
```

ونفس الأمر بالنسبة لباقي الحلقات التكرارية While و for each وخلافه .

* تاسعاً : بالنسبة للمصفوفات وخلافه يستخدم القوس [] بدلاً من () في VB.net .

كود C#

```
int[] nums = { 1, 2, 3 };
for (int i = 0; i < nums.Length; i++)
    Console.WriteLine(nums[i]);
```

كود VB.NET

```
Dim nums() As Integer = {1, 2, 3}
For i As Integer = 0 To nums.Length - 1
    Console.WriteLine(nums(i))
Next
```

- لن تجد تعليقاً Comment لأكثر من سطر في VB.net مثل /* */ في C# ، وكذلك XML Comments على الرغم من أنني قرأت انها قد تكون مدعومة في الاصدارة القادمة .

- في C# سوف تستخدم العلامة المئوية % بدلاً من Mod في VB.net للحصول على باقي القسمة.

- لن تدعم لك VB.net استخدام Bitwise Operations في حالة Assignment على عكس ال C# حيث تتيح لك ذلك .

لن اطيل لإنك - كما لاحظت معي - فإن هذه الفروقات لا تتعدى كونها فروقات لغوية ، وهي فروقات غير ذات قيمة بالمرة ، ساحيلك على رابط لعرضها كاملة في نهاية الدرس ، ولكن الآن لنحاول استعراض النقاط غير تلك المهتمة بأسلوب الكتابة .

- أول تلك الفروقات هي case sensitive ، حيث أنه في VB.net فإن Ahmed هي نفسها ahmed وهو ما لا يوجد في أي لغة في العالم سوى Basic ، ولا أستطيع أن أحدد فيما إذا كان هذا ميزة أم عيب.

مميزات في VB.net - باختصار -

- دعم ال optional parameters موجود في VB.net وليس موجود في C# .

- with موجود في VB.net وليس موجود في C# .

- When ... Catch موجودة في VB.net وهي تتيح نظاماً أفضل لفلترة الأخطاء

- يقوم VB.net بعمل Compile للكود في BackGround ، وهذه ميزة في التطبيقات الصغيرة ولكنها عيب في التطبيقات الكبيرة حيث تلاحظ بطء الفيجوال ستوديو .

مميزات في C# - اختصار - :

- يدعم C# ما يعرف باسم unsigned types ، وهو ما يمكنك استخدامه أيضاً في VB.net ولكنه ليس جزءاً اساسياً من اللغة .

- الميزة الأقوى في C# والتي لا تتوفر في VB.net هي السماح باستخدام

unsafe أو unmaneged code ما يتيح لك العمل على Pointers وخلافه ، ما يفتح لك آفاقاً واسعة في عدة مجالات مثل معالجة الصور image processing وخلافه .

ولعل هذه الميزة مما جعل C# قريبة من C++ من ناحية تعاملها مع كود منخفض المستوى Low Level Code وهو ما يجعل سي شارب الاختيار الأمثل للكثيرين .

- لن تجد increment و decrement في VB.net حيث ستتضطر إلى كتابة كود بالشكل التالي :

كود	VB.NET
	A=A+1 A-=1

بينما في C# يمكنك القيام بذلك بالشكل التالي :

كود	C#
	A++ A–

- يمكنك أن تجد الخاصية sizeof في C# بينما لن تجدها في VB.net.

كان هذا موجزاً لأوضح الاختلافات والتي ستتعامل معها كثيراً ، وكما لاحظت فهي في أغلبها ليست ذات قيمة كبيرة .

في النهاية : ماذا أختار ؟

إذا كنت منتقلاً من VB 6.0 فعليك ب VB.net أما إذا كانت لديك خبرة ب C/C++/Java فعليك ب C# وإذا كنت جديداً في مجال البرمجة فأنت حر في اختيار ما يناسبك وإن كنت أرشح لك C# كرأي شخصي فقط .

هذا الملف من موقع مايكروسوفت لتوضيح كامل للفروقات :

رابط

http://support.microsoft.com/kb/308470

4. مقدمة إلى Mono

في عالم الجافا تعتمد على VM والذي يجعلها قابلة للعمل على اي مكان عليه VM حتى لو كان ثلاجة او فرن مايكروويف ، في المقابل يسبب هذا الموضوع بعض البطء في تنفيذ الجافا ولكنه على كل حال من أهم مميزات الجافا.

وفي العالم المقابل عالم Microsoft كانت منتجات Visual Basic و C# أو MFC غير قابلة للعمل سوى على نظم التشغيل من مايكروسوفت.

ولكن مع الوقت تم تقدم طرق وأدوات لتسمح لبرامجك بالعمل على نظم تشغيل متعددة ، أشهرها هو Mono وهو موضوع درسنا الحالي.

في الواقع حتى اللحظة لم أجد مصدر يفيد بأن Mono اصبحت متوافقة مع.net 2008، ولكنها متوافقة مع .net 2.0 سواء على شكل ASP.net أو برامج تطبيقية أو قواعد بيانات وخلافه ، وبرغم ذلك وجدت عدة روابط للمحاولات الحالية للتوافق تفيد بوصولهم إلى تطبيق أغلب نقاطها - ليس كلها - ، على كل يمكنك الدخول على هذا الرابط لمتابعة الاخبار:

رابط

http://www.mono-project.com/plans

من الموقع السابق قم بعمل Download لل Mono ، وخلال مراحل التحميل قم باختيار التقنيات التي ترغب ل Mono بالعمل معها.

المرحلة الثانية هي بتشغيل ال Command Line الخاص بال Mono والذي ستجده في قائمة البرامج ، ستجد من ضمن الأدوات المتاحة لك الأدوات التالية:

mcs/gmcs: كومبايلر للسي شارب

Vbnc: كومبايلر للفيجوال بيسك

ilasm/ilasm2:كومبايلر للCIL

كل واحدة منهم - السي شارب كمثال - يتم التعامل معها كما قمنا بالتعامل سابقاً مع ال Visual Studio Command Line سابقاً ، الفارق الوحيد بين gmcs و mcs هو أنgmcs يدعم التقنيات الجديدة الموجودة مع عالم.net 2.0 .

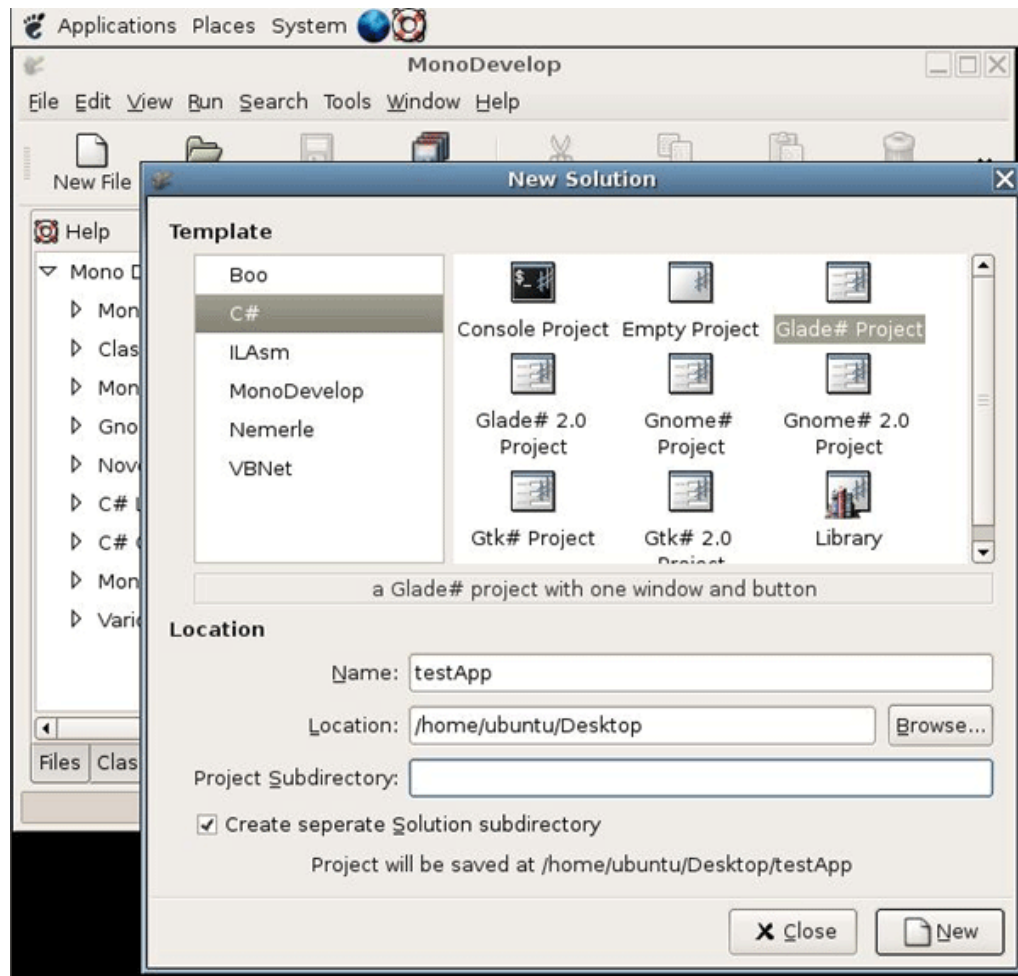
لنفترض ملف باسم sample.cs يحتوي على كود C# عادي جداً ، قم بفتح الCommand Line الخاص ب Mono ثم اكتب الأمر التالي:

كود	Shell

```
gmcs /target:library /out:Sample.dll Sample.cs
```

وفقط!

الآن اصبحت لديك مكتبة قابلة للعمل على اي نظام تشغيل ، يمكنك عمل build ك exe ايضاً وتشغيلها عبر mono في لينكس مثلاً...

على اللينكس يمكنك فتح مشروعك باستعمال MonoDevelop كما على الVISUAL STUDIO :

الصورة 21. 9. اضافة مشروع c# جديد في ال MonoDevelop على اللينكس.

5. برمجة الأجهزة الكفية من خلال ال net.

آخر مواضيعنا المتقدمة قبل الولوج لعالم الويب هو برمجة الأجهزة المحمولة ، ولكن قبل البداية،

ما هي الأجهزة الكفية Pocket PC ؟

هي مجموعة من أجهزة الهواتف المتنقلة تتميز باستخدام نظام تشغيل متقدم اضافة إلى هاردوير يمكنها من تشغيل مهام هذا النظام ، ويشكل نظاما التشغيل Windows Mobile ونظام

التشغيل سيمبان اضافة لنظام Windows CE 2003 وما قبله أشهر نظم التشغيل التي تعمل على الأجهزة الكفية .

أما على صعيد الهاردوير فالأجهزة الكفية لا تمتلك هارد ديسك Hard Disk بالمعنى المشهور ، لكن بعضاً منها يستخدم جزء من ال RAM كهارد ديسك لا يتم مسح البيانات من عليه بعد اعادة تشغيل الجهاز ، وهناك أنوع أخرى تستخدم ROM لتخزين البرامج ونظام التشغيل وما شابه .

وماذا يمكنني عمله للأجهزة الكفية Pocket PC من خلال .net ؟

تستطيع من خلال لغة البرمجة اضافة برامج تستخدم أجزاء معينة من الجهاز لتحقيق خدمات للمستخدم سواء أكان استخدام بسيط (استخدام الذاكرة) لتخزين البرامج وتنفيذها وتخزين قواعد البيانات ، أو كان الأمر يختص باستخدام الكاميرا مثلاً والأجهزة الصوتية في الجهاز .

كيف أبدأ ؟

بكل بساطة ، سنقوم بعمل برنامج يظهر رسالة ترحيب عن الضغط على زر أمر .

1- قم بتشغيل الفيجوال ستوديو Visual Studio .net .

2- قم باختيار VB أو اللغة التي تود البرمجة من خلالها .

3- قم باختيار Smart Device ، ومن ثم قم باختيار Pocket PC 2003 أو Smartphone 2003 أو Windows CE 5.0 حسب الجهاز الذي تود العمل عليه .

4- بعد اختيار القسم المناسب قم باختيار Device Application .

5- قم بسحب Button من الأدوات ، وقم بكتابة هذا الكود في داخله

كود	C#

```
MessageBox.Show("Welcome...");
```

كود	VB.NET

```
MessageBox.Show("Welcome...")
```

6- قم بالضغط على F5 اختار نوع ال deploy الذي ترغب فيه حسب الجهاز الذي تود العمل عليه ، أو قم باختيار نوع يعمل على الـ PC للتجربة فقط .

7 - مبروك ، لقد قمت بعمل برنامج الأول ، يمكنك أيضاً نقل الملف التنفيذي exe إلى الجهاز وتشغيله مباشرة ، لا بد في هذه الحالة ان يكون net compact framework. موجوداً على الجهاز الكفي Pocket PC .

8- المهام الأساسية – غير تلك التي تعودت عليها في البرمجة التقليدية هي تلك المختصة بالتعامل مع بنية الجهاز وخلافه ، فمثلاً لتحديد اتجاه الشاشة في الأجهزة التي يمكن أن تنقلب فيها الشاشة :

كود	C#

```
SystemSettings.ScreenOrientation = ScreenOrientation.Angle90;
```

كود	VB.NET

```
SystemSettings.ScreenOrientation = ScreenOrientation.Angle90;
```

ولتمكين Touch Keyboard :

كود	C#

```
InputPanel1.Enabled = true;
```

كود	VB.NET

```
InputPanel1.Enabled = True
```

ومثلاً لعمل Reset للجهاز بعد استيراد coredll.dll :

كود C#

```
private int CTL_CODE(int DeviceType, int Func, int Method, int Access)
{
    return (DeviceType << 16) | (Access << 14) | (Func << 2) | Method;
}
private int ResetPocketPC()
{
    const int FILE_DEVICE_HAL = 257;
    const int METHOD_BUFFERED = 0;
    const int FILE_ANY_ACCESS = 0;
    int bytesReturned = 0;
    int IOCTL_HAL_REBOOT;
    IOCTL_HAL_REBOOT = CTL_CODE(FILE_DEVICE_HAL, 15, METHOD_BUFFERED,
FILE_ANY_ACCESS);
    return KernelIoControl(IOCTL_HAL_REBOOT, IntPtr.Zero, 0, IntPtr.Zero, 0,
bytesReturned);
}
```

كود VB.NET

```
Private Function CTL_CODE(ByVal DeviceType As Integer, ByVal Func As Integer,
ByVal Method As Integer, ByVal Access As Integer) As Integer
    Return (DeviceType << 16) Or (Access << 14) Or (Func << 2) Or Method
End Function
Private Function ResetPocketPC() As Integer
    Const FILE_DEVICE_HAL As Integer = &H101
    Const METHOD_BUFFERED As Integer = 0
    Const FILE_ANY_ACCESS As Integer = 0
    Dim bytesReturned As Integer = 0
    Dim IOCTL_HAL_REBOOT As Integer
    IOCTL_HAL_REBOOT = CTL_CODE(FILE_DEVICE_HAL, 15, METHOD_BUFFERED,
FILE_ANY_ACCESS)
    Return KernelIoControl(IOCTL_HAL_REBOOT, IntPtr.Zero, 0, IntPtr.Zero, 0,
bytesReturned)
End Function
```

كيف اكمل البرمجة من خلال .net ؟

بكل تأكيد فإن البرامج السابقة يعد من أبسط الصيغ الممكنة لبرنامج يعمل على Pocket PC، فيما ستتضطر لاستخدام التخزين في البرامج الجدية والتعامل مع قواعد البيانات حيث يمكنك استخدام الملفات النصية كقواعد بيانات ، أو ملفات XML وحتى قواعد البيانات من نوع SQL Server CE .

يمكنك الدخول أيضاً مباشرة على هذا الرابط من مايكروسوفت لتبدء منه :

رابط

http://msdn2.microsoft.com/en-us/library/aa458721.aspx

الباب

22

البرمجة باستخدام ال

ASP.NET

1. مقدمة إلى تطوير المواقع

فيما مضى من الدروس ، كنا نتحدث في عالم ال Console والDesktop Applications، وخلال هذه المراحل كنا نعتمد على وجود جهاز واحد للتنفيذ ، وحتى في حالة وجود شبكة فقد كان الوضع يعتمد على عمليات ارسال واستقبال بيانات بين الجهازين او المشاركة في المصادر او الاتصال ب web service كما رأينا ، اما مع عالم ال ASP.net او عالم الserver side languages عموماً فالوضع مختلف نسبياً ، ولذا قبل البداية نحب ان نتعرف على انواع مواقع الإنترنت:

- مواقع ثابتة : هذه المواقع عبارة عن مجموعة من التوصيفات باستخدام HTML، يمكن استخدام صور وفلاشات وخلافه ولكن لا يوجد اي نوع من انواع المعالجة في هذا الموقع.

- موقع ديناميكية عند العميل : هذا النوع من المواقع يتمتع ببعض انواع المعالجة ولكنها تظل في جانب العميل فقط ، يتم في هذه المواقع استخدام Scripts مثل JavaScript و VBScript ، لكن الصفحة التي تراها امامك تعمل عندك انت فقط.

- مواقع دينامكية : هذا النوع يتم عمل معالجة لبياناته في السيرفر وتحصل انت فقط على الناتج الذي يمكن ان يكون من النوع الأول والثاني ، في هذه الحالة يسمى تطبيق انترنت وليس موقع انترنت.

مثال:

صفحة بريدك الالكتروني التي تراها امامك هي صفحة من النوع الثاني حيث تحتوي على بعض اوامر الجافا سكريبت اضافة لبعض الجداول والحقول وخلافه من HTML ، لكن في الواقع فإن تشكيل هذه الصفحة بهذه الصيغة جاء عن طريق بعض عمليات السيرفر التي قامت بقراءة بعض قواعد البيانات وشكلت لك الجداول التي تحتوي على البريد الوارد لك وخلافه.

لذا في هذه الصفحة انت غير قادر على رؤية اكواد اللغة التي تمت برمجة السيرفر بها ، ولكنك تستطيع مشاهدة النتائج فقط حيث ان عمليات المعالجة تتم على السيرفر فقط.

دورة حياة تطبيق الإنترنت:

بمجرد ادخالك لعنوان موقع ما يبتدأ ب - //:http وهو ما يعني انك ستستخدم بروتوكول http - يقوم البروتوكول بارسال هذا العنوان إلى ما يعرف بDomain Name Server واختصاراً باسم DNS ، يقوم هذا ال DNS بتحويل الكتابة التي قمت بكتابتها لعنوان Physical على الإنترنت خاص بال IP صاحب الموقع ، شكل ال DNS يكون بالشكل التالي كمثال:

Name	IP
www.yahoo.com	8 bit – 4 parts IP
www.hotmail.com	xxxxxxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx

الصورة 22. 1. عملية تحويل العناوين النصية إلى عناوين IP في خوادم ال DNS.

الآن اصبح بالامكان معرفة IP المميز لهذا الموقع ، الآن يتم فتح - port غالباً ما يكون 80 - ويتم ارسال طلب المعالجة إلى هذا الموقع وهو ما نطلق عليه http Request .

في حالة كون الموقع static، يتم الرد بالصفحة المطلوبة فقط ، اما في حالة كون الصفحة dynamic يتم معالجة البيانات المطلوبة وارسال صفحة النتائج إلى المستخدم وهو ما نطلق عليه http Response .

- لكل بروتوكل دورة عمل ، وما يهمنا هنا هو http فقط.

مواقع العميل Client Side Web Sites

يستلزم هذا النوع بداية معرفة ب HTML ، وهي لغة سهلة جداً يمكنك التعرف عليها خلال اقل من نصف ساعة عن طريق هذا الرابط مثلاً:

رابط

www.html4arab.com

ومع ظهور برامج ابتداء ب Front Page وانتهاء ب Dream Waver ، اصبح بامكانك عمل الموقع دون اي معرفة بال HTML عن طريق الادوات فقط.

ايضاً تحتاج ل Java او vbscript لبرمجة بعض الخصائص ، ايضاً ضمن ال HTML تجد عالم ال Forms الذي سيمكنك من التواصل مع العالم الخارجي ، وتحتاج لبعض المعرفة في هذا الموضوع، يمكنك مراجعة الدرس التالي كبداية للvbscript – من مدونتي - :

رابط

http://ahmedgamal-technical.blogspot.com/2008/08/vbscript.html

والدرس التالي كبداية ل javascript :

رابط

http://www.w3schools.com/JS/default.asp

والآن اتوقع انك ملم بالاساسيات فقط ، لا اطلب منك الكثير في هذا المجال حالياً...

الفرق بين تطبيق الويب Web Application وخدمة الويب Web Service

- تطبيق الويب هو عبارة عن مجموعة من الصفحات وملفات الاكواد وقواعد البيانات التي لها دورة حياة خاصة مثلها مثل اي برنامج ، يتم فتحها واغلاقها وتنفيذ احداث وخلافه

- اما خدمة الويب فهي عبارة عن خدمة تستقبل بعض البيانات وتعيدها بعد بعض عمليات المعالجة، يتم التعامل معها من اي لغة برمجة قادرة على الوصول إلى اسمها ورقم البورت فقط.

ال IIS :

هو السيرفر الخاص ببرمجيات ASP و ASP.net ، ووجوده شرط اساسي ليعمل كودك الASP.net سواء على جهازك الشخصي أو على السيرفر الذي ستسضيف عليه موقعك لاحقاً.

كونك قمت بعمل setup ل visual studio يعني انك قمت بتحميل ال IIS او للدقة فأنت قمت بتحميل WebDev.WebServer.exe والذي يشمل ال IIS ، اما إذا كنت تود البرمجة على المفكرة Notepad مثلاً فلا بد من عمل setup له ، وذلك عن طريق اختيار لوحة التحكم Control Panel

اضافة وازالة برامج Add And Remove Programmes واختر اضافة مكونات ويندوز Add Windows Compomnent ، ستجد من ضمنها Internet Inforamtion Services وهو الIIS.

** الطريقة السابقة كل حسب نظام التشغيل الذي يعمل عليه.

تجارب بسيطة

قم بفتح Notepad جديد ، قم بتسميتها لتكون index.html مثلاً ثم قم بكتابة المحتويات التالية فيها:

كود | HTML

```html
<html>
    <head>
        <title>My First Pgae</title>
    </head>
    <body>
        <center><img src="http://l.yimg.com/a/i/ww/beta/y3.gif"></centeR>
        <br>
        <font color=red size=3>Yahoo!</font>
        <br>
        <a href="http://www.yahoo.com">Link Here</a>
    </body>
</html>
```

جرب تشغيلها مباشرة بالضغط المزدوج عليها، مبروك ، هذه اول صفحة انترنت لك.

الآن سنقوم برسم الفورم التالي لادخال البيانات:

Welcome !!

Name

Password

do anything   Clear

الكود الخاص بهذا الفورم كالتالي:

كود	HTML

```
<html>
    <head>
        <title>My First Pgae</title>
    </head>
    <body>
        <h2>Welcome !!</h2>
        <br>
        <form name="form1">
            <table border=0>
                <tr>
                    <td>Name</td>
                    <td>
                    <input type=text name=textname size=10>
                    </td>
                </tr>
                <tr>
                    <td>Password</td>
                    <td><input type=password name=textpass size=10><td>
                </tr>
            </table>
            <br>
            <input name="button1" type=button value="do anything"();">
               
            <input type=reset value="Clear">
        </form>
    </body>
</html>
```

والآن نريد ان نقوم بالتأكد من أن المستخدم قام بادخال بيانات في الفورم ، لو قام بادخال بيانات فسوف نظهرها له في رسالة نصية وعلى الشاشة ايضاً ، لذا سنذهب للزر Button1 ونغيره ليصبح بالشكل التالي:

كود	HTML

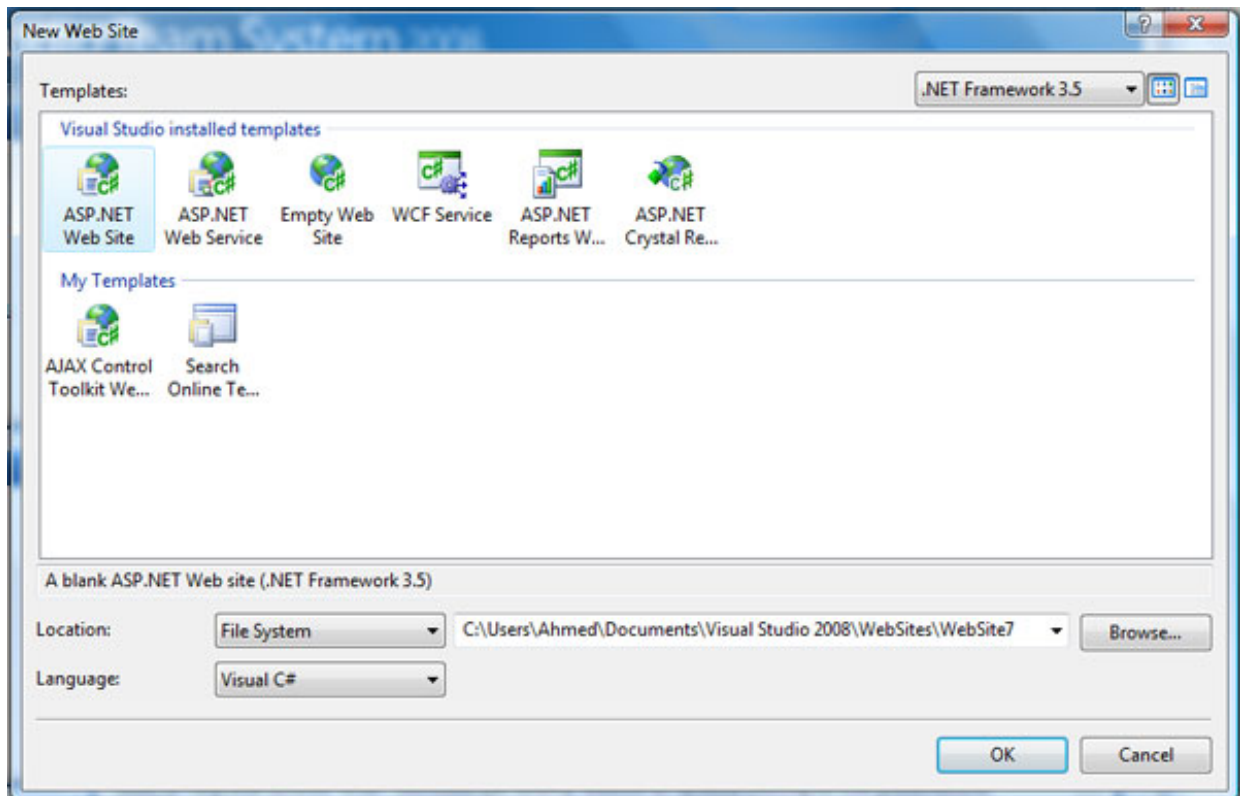
```
<input name="button1" type=button value="do anything" onClick="doCheck();">
```

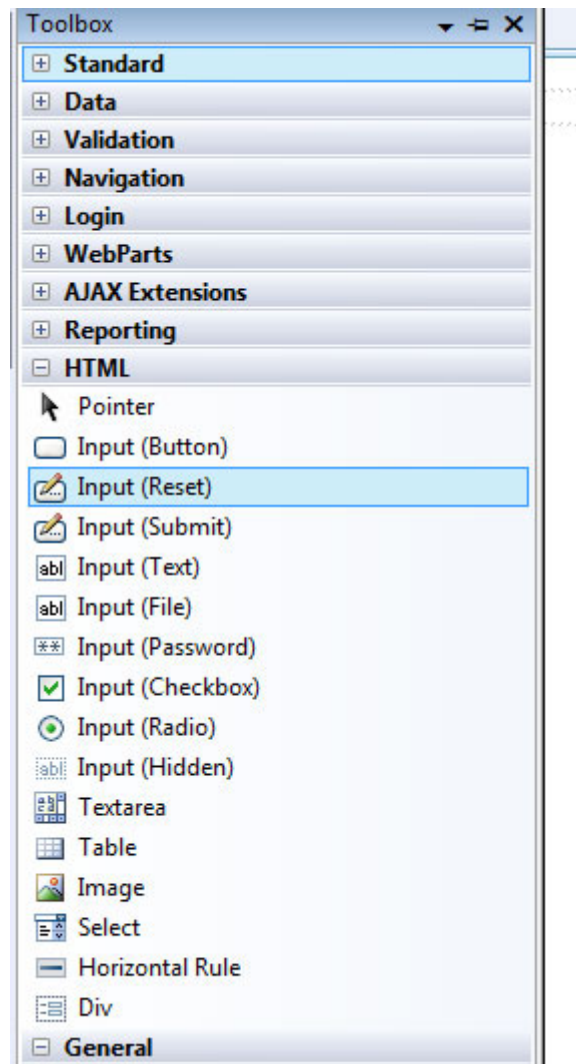
ومن ثم في آخر الصفحة سوف نقوم بكتابة ال Script التالي:

كود	HTML

```
<script language="javascript">
    function doCheck()
    {
        if(form1.textname.value!="" || form1.textpass.value!="")
        {
            alert(form1.textname.value);
            document.write(form1.textpass.value);
        }
        else
        alert("enter data first !");
    }
</script>
```

استخدام Visual Studio

سننقل نفس هذه التجربة البسيطة ولكن باستخدام ASP.net من خلالVisual Studio، قم بفتح الفيجوال ستوديو وقم باختيار ويب جديد Website ثم اختر ASP.net Web Site

الصورة 22. 2. اضافة مشروع جديد (موقع ويب بال ASP.NET)

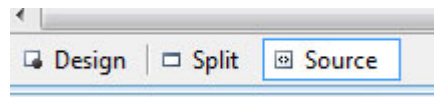
الآن قم برسم نفس الفورم ولكن من الادوات الجانبية:

الصورة 22. 3. أدوات ال HTML في الفيجوال ستوديو.

ملاحظة

لا تنس اننا حتى اللحظة نتعامل مع ادوات HTML وليست ادوات ASP.net

وعندما تقوم بوضع الكود ، قم بالضغط المزدوج على Button وقم بكتابة الأوامر مباشرة:

الصورة 22. 4. محرر ال HTML المتقدم في تطبيقات ال ASP.NET

يمكنك استعراض التصميم والكود سوية او كل واحدة منهما عن طريق التبويب اسفل الصفحة:

فقط ... كانت هذه هي تجربتنا البسيطة ، في الدرس القادم سنبدأ بالتعامل مع ASP.net .

2. مقدمة إلى ASP.net

في درسنا السابق جربنا التعامل مع Visual Studio وعمل صفحات ويب غير تفاعلية ، في هذا الدرس سنجرب تجربتنا الأولى مع المواقع التفاعلية.

تجربة ASP.net

قم بفتح الأدوات Standerd، قم برسم مربع نص TextBox واداة عنوان Label وزر أمر Button، في زر الأمر قم بكتابة الكود التالي:

كود	C#

```
Label1.Text= TextBox1.Text;
```

كود	VB.NET

```
Label1.Text= TextBox1.Text
```

وجرب ... في الواقع لقد قمت بعمل اول صفحة ASP.net لك.

ماذا حدث فعلياً ؟

لو فتحت صفحة ال HTML ستجد أن لديك كود HTML عادي يحتوي على فورم ، في الواقع فإن ما تم هو حدوث Submit إلى السيرفر حيث قام بارسال بيانات الفورم كاملة إلى السيرفر ، وهناك على السيرفر تمت معالجة البيانات وتم اعادة الصفحة التي تحتوي على Label يحتوي على المحتوى Ahmed او اياً كان محتوى مربع النص قبل الارسال ، لو جربت مشاهدة الكود HTML الخاص بالصفحة الناتجة ستجد الجزء التالي من الكود:

كود	HTML

```
<p>
    <span id="Label1">Ahmed</span>
</p>
<p>
    <input name="TextBox1" type="text" value="Ahmed" id="TextBox1" />
</p>
<p>
    <input type="submit" name="Button1" value="Button" id="Button1" />
</p>
```

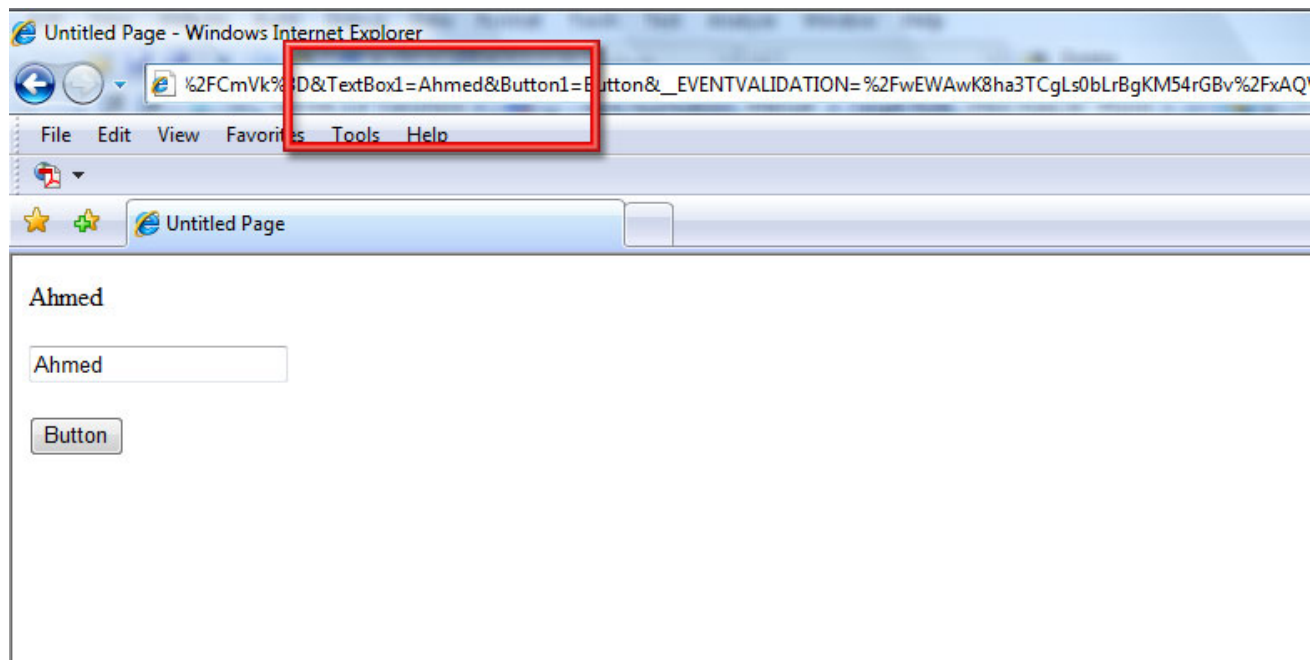
كما ذكرنا ، تمت معالجة البيانات ، وتم اعادة الناتج إلى Label1، اما الاكواد وخلافه فهي في السيرفر فقط.

انواع ارسال البيانات.

في أي Form هناك طريقتين لارسال البيانات هي POST و GET ، في الطريقة الأولى وهي الافتراضية يتم ارسال بيانات ال Form مباشرة، اما في GET فيتم ارسالها في عنوان المتصفح، جرب تعديل الفورم الخاص بنا ليصبح بالشكل التالي:

كود HTML

```
<form id="form2" runat="server" method="get">
```

قم بتجربة الموقع مرة أخرى ، لاحظ الصورة التالية:

الصورة 22 .5. استخدام ال GET Method لارسال البيانات.

كما ترى ، يتم ارسال كافة محتويات الفورم في العنوان ، طبعاً معظم المتصفحات تضع حدوداً على طول الفورم أما في حالة POST فليست هناك اي قيود.

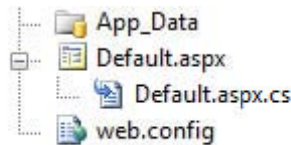
لكل واحدة منهم طريقة قراءة فيما لو اردت قراءتها منفصلة بالكود ، وهو ما قد نتعرف عليه في مرات قادمة إن شاء الله تعالى.

Code Behind

من ضمن المميزات الجديدة التي ظهرت مع ASP.net عن ASP القديمة هي فصل الكود عن التصميم ، في صفحة ASP Classic قديمة كنت ترى هذا الكود مثلاً:

كود ASP.NET

```
<%
    v_Id = Request.QueryString("id")
    Response.Write("<form action=test7.asp?id=" & v_id & " method=post>")
    Response.Write("<font color=red><b><center>Write your
C.V.</center></b></font><br>")
%>
    <table>
        <tr>
            <td>Computer Skills :</td>
            <td><input type=text name=v_cs size=30></td>
        </tr>
        <tr>
            <td>Characteristics :</td>
            <td><input type=text name=v_char size=30></td>
        </tr>
        <tr>
            <td>Interest's :</td>
            <td><input type=text name= v_inter size=30></td>
        </tr>
        <tr>
            <td>Language Skills :</td>
            <td><input type=text name=v_ls size=30></td>
        </tr>
        <tr>
            <td>Education :</td>
            <td><textarea name=v_edu cols =30 rows=6>write your Study Field
here</textarea></td>
        </tr>
        <tr>
            <td>Experince :</td>
            <td><textarea name=v_exp cols =30 rows=6>write your Experiences
here</textarea></td>
        </tr>
    </table>
    <br>
    <hr width=75%>
    <center>
        <font color=red><b><center>Join Demand</center></b></font><br>
        <textarea name=v_Join cols=40 rows=9 WRAP=physical></textarea>
        <br><br>
        <input type=submit value= Continue>
        <input type=reset value=Erase>
    </center>
</form>
```

أما في ASP.net فاصبحت أكوادك في ملف بامتداد cs.* في حين اصبح التصميم في ملف منعزل بامتداد aspx.* :

وبرغم ذلك ما زلت قادراً على دمجهم في ملف واحد إن احببت ذلك وهي طريقة ما زال يفضلها الكثيرين ، وايضاً قد تحتاج في بعض الاحيان لتطعيم ملف ال aspx ببعض اكواد ال ASP خصوصاً في المشاريع المتشعبة.

بخصوص اخفاء الكود ، فعلاً صمم لعدم عمل تداخل ولتسهيل الامر على المبرمج ، ولكن برغم ذلك يظل اسلوب الكتابة في صفحة واحدة انسب لعدة نقاط:

- احياناً ما تحتاج لوضع بعض الاكواد التي لا تستطيع العمل عليها من خلال صفحة - cs.* والذي يعمل في مشروع كبير يستطيع ان يدرك ذلك. -

- اسهل في التوزيع والنقل بين المبرمجين.

- وفي نفس الوقت اسهل في التعديل ، فتعديل اسم حقل يتطلب التعديل في ملف واحد فقط - رغم ان Visual Studio يقدم لك ميزة التعديل المباشر إلا ان عواقبها وخيمة عن تجربة.

- وكذلك هي اسهل في عملية ال Deploy لعدم وجود ملفات معتمدة على بعضها البعض وهو موضوع درسنا التالي.

مكونات مشروع ال ASP.net

لو قمت بتصفح ال Website Directory الموجود على يمين المتصفح ، ستجد انه يحتوي اضافة لملفات مشروعك على المجلدات التالية - حتى لو لم تكن موجودة فهذه الاسماء القياسية لكتابتها :-

App_Browsers : يحتوي على الملفات التي تقتنص نوع المتصفح وتتعامل مع كل متصفح بناء على امكانياته.

App_Code : يحتوي على ملفات الاكواد والفئات Classes التي تخص صفحات الويب.

App_Data : يحتوي على ملفات قواعد البيانات.

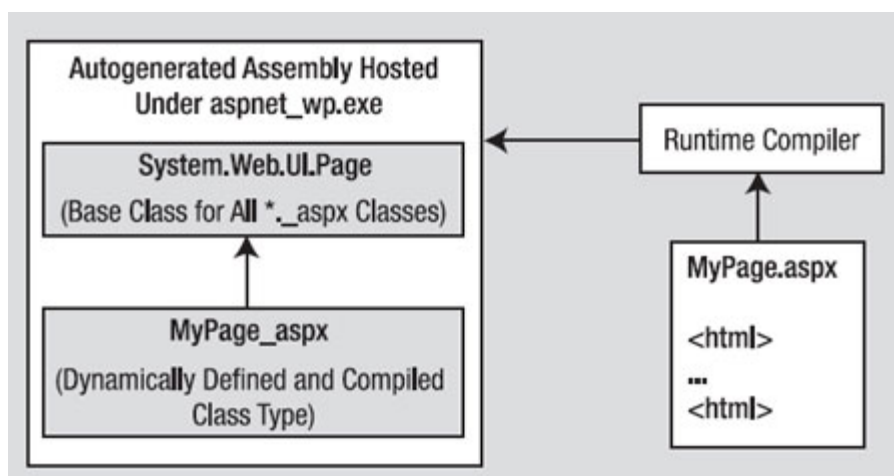
App_GlobalResources : تحتوي على ملفات المصادر *.resx .

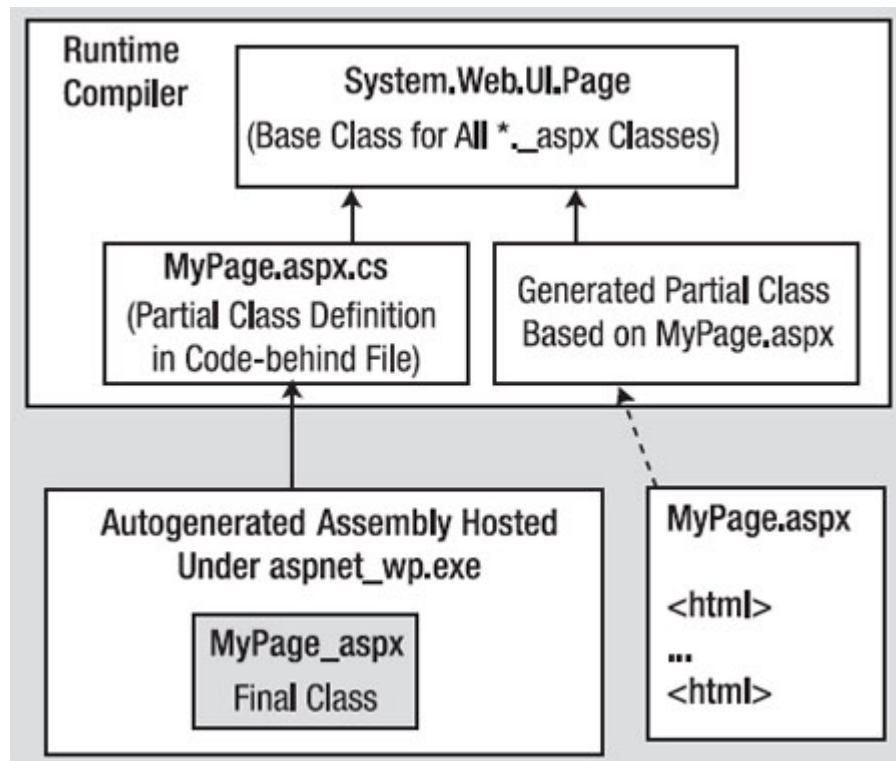
App_Themes : يحتوي على الملفات التي تعني بالمظهر والستايلات الخاصة بالمشروع.

App_WebReferences : في حالة وجود بروكسي Proxy اوي اي نوع من الWeb Service يستخدمها تطبيقك يتم وضعها هنا.

Bin : يحتوي على الملفات الجاهزة مثل ملفات ال *.dll والتي يتم استخدامها في برنامجك.

دورة الترجمة Compilation Cycle

في حالة كون الملف Single File ، يتم ترجمة الملف بكامل محتوياته إلى فئة Class مشتقة من System.Web.UI.Page باسم نفس الصفحة اضافة ل _aspx.

الصورة 22 .6. دورة ترجمة تطبيقات ال ASP.Net.

أما في حالة استخدامك لل Mode الخاص بعمل Code Behind ، فسيتم عمل نفس الخطوات ولكن بدمج ثلاث ملفات سوية، الملف المسمى الذي يحتوي على ال InitializeComponent() يتم عمل Compile له ، ليتم لاحقاً اشتقاق الملف المتكون من ASPX + CS منه وليس من System.Web.UI.Page مباشرة.

الصورة 22. 7. دورة ترجمة صفحات ال ASP.NET.

*** الصور منقولة...

3. الفئة System.Web.UI.Page

تحتوي هذه الفئة على مجموعة من الخصائص والفئات التي ستفيدك في عملك، منها:

الفئة	الوصف
Application	التعامل مع متغيرات التطبيق وما يختص بالموقع
Cache	للتعامل مع الكاش الخاص بهذا الموقع
IsPostBack	لمعرفة هل الصفحة يتم عمل Load لها من جديد أم انه يتم عمل Load لها بناء على عملية Submit حدثت من الصفحة
MasterPageFile	تحديد الصفحة الماستر ، سنتعرف عليها لاحقاً
Request	ال HttpRquest ، سنتعرف عليها في الدرس التالي
Response	ال HttpResponse، سنتعرف عليها في الدرس التالي
Server	الوصول إلى الدوال الخاصة بالتعامل مع السيرفر ضمن HttpServerUtility
Session	التعامل مع ال Session، تستخدم لتخزين بعض القيم وسنتعرف عليها لاحقاً
Theme	لتحديد الثيم - لا اعرف ترجمة له - الخاص بهذه الصفحة

الجدول 22. 1. أهم الفئات في مجال الأسماء System.Web.

3. 1. التعامل مع ال Request

عملية ال Request هي عملية ارسال بيانات إلى السيرفر لتنفيذ مهمة ما عليها ، عملية ارسال البيانات يطلق عليها اسم Request.

مثال : بعد ادخال البيانات الشخصية تقوم بالضغط على زر (ارسال) هذا الزر يقوم بمهمة Request.

في العادة ، يتم تعريف ال Action وهي الصفحة التي يتم ارسال البيانات إليها ، كما يتم تحديد اسلوب الارسال Get أو Post التي شرحناها سابقاً وذلك في تعريف الفورم بالشكل التالي مثلاً:

كود	HTML

```html
<form name="form1" id="form1" action="Process.asp" method = "GET">
...
</form>
```

في net. نفس النظام ايضاً ، ولكن يتم تعريف الفورم بطريقة مختلفة قليلاً:

كود	HTML

```html
<form id="form1" runat="server">
```

هذا يعني افتراضياً ان الصفحة التي سيعود فيها هي نفس الصفحة كما ان طريقة الارسال الافتراضية هيPOST.

غير ان ASP.net لا تتيح لك الوصول مباشرة إلى HttpRequest، ولكنها تمنحك هذه الخاصية ضمن System.Web.UI.Page.Request والتي تجد فيها الدوال والخصائص التالية:

الدالة-الخاصية	الوصف
ApplicationPath	مسار الموقع على السيرفر
Browser	نوع متصفح المستخدم، وهي فئة تحتوي على عدد كبير من المعلومات
Cookies	معرفة ملفات الكوكيز التي تم ارسالها من قبل المستخدم
HttpMethod	نوعية الارسال Set اوGet
Form	محتويات الفورم الذي تم ارساله للسيرفر في حالة كون الارسالPost
QueryString	محتويات الفورم الذي تم ارساله للسيرفر في حالة كون الارسال Get ، حيث يقوم بقراءة محتويات ال URL مباشرة حتى

	لو لم تكن من ضمن محتويات الفورم
IsSecureConnection	لمعرفة هل يتم تطبيق اتصال Http آمن ام لا
RawUrl	معرفة ال URL بدون اي اضافات
ServerVariables	الوصول والتعامل مع مجموعة متغيرات السيرفر
UserHostAddress	معرفة IP العميل
UserHostName	معرفة اسم المستضيف للعميل
MapPath()	دالة تقوم بتحويل المسار المطلوب إلى مسار حقيقي على السيرفر
SaveAs()	حفظ محتويات ال http على ملف على السيرفر

الجدول 22. 2. أهم دوال وخصائص الفئة Request

وسنستعرض لبعض الامثلة عن استخدام Request...

قراءة بيانات التي تم ارسالها في Form

لو كنت تستخدم اسلوب Post:

كود | C#
```
firstName = Request.Form("txtFirstName");
```

كود | VB.NET
```
firstName = Request.Form("txtFirstName")
```

ولو كنت تستخدم اسلوب: GET

كود | C#
```
firstName = Request.QueryString["txtFirstName"];
```

كود	VB.NET

```
firstName = Request.QueryString("txtFirstName")
```

وبرغم انك تستطيع قراءتها مباشرة باستخدام ال ID كما شرحنا في أول الدروس ، إلا انك ستحتاج لهذه الطريقة في عمليات أخرى اضافية.

معرفة احصائيات المتصفح المرسل

كود	C#

```
string theInfo = "";
string isAOL = string.Format("Is AOL? {0} <br/>", Request.Browser.AOL);
string isActivex = string.Format("Support ActiveX? {0} <br/>",
Request.Browser.ActiveXControls);
string isBeta = string.Format("Is Beta? {0} <br/>", Request.Browser.Beta);
string isJava = string.Format("Support Java Applets? {0} </br>",
Request.Browser.JavaApplets);
string isCookies = string.Format("Support Cookies? {0} <br/>",
Request.Browser.Cookies);
string isVB = string.Format("Support VBScript? {0} <br/>",
Request.Browser.VBScript);
```

كود	VB.NET

```
Dim theInfo As String = ""
Dim isAOL As String = String.Format("Is AOL? {0} <br/>", Request.Browser.AOL)
Dim isActivex As String = String.Format("Support ActiveX? {0} <br/>",
Request.Browser.ActiveXControls)
Dim isBeta As String = String.Format("Is Beta? {0} <br/>",
Request.Browser.Beta)
Dim isJava As String = String.Format("Support Java Applets? {0} </br>",
Request.Browser.JavaApplets)
Dim isCookies As String = String.Format("Support Cookies? {0} <br/>",
Request.Browser.Cookies)
```

3. 2. التعامل مع ال Response

العملية العكسية لعملية Request، حيث تمثل عملية نقل البيانات من السيرفر إلى العميل مرة أخرى ، حيث تشكل الناتج الذي سيتم ارساله للمستخدم ، وتحتوي على الخصائص والدوال التالية:

الخاصية أو الدالة	الوصف
ContentEncoding	نظام الترميز المستخدم في الناتج
Cookies و Cache	كما في حالة ال Request ولكن في العملية العكسية
IsClientConnected	للتأكد من استمرارية اتصال المستخدم حتى اللحظة
Clear()	مسح كافة محتويات ال Body والHeaders
End()	انهاء عمليات المعالجة وارسال ما تم انهاءه فقط
Flush()	ارسال ما تم انجازه للعميل دون ايقاف عملية المعالجة ** مفيدة جداً **
Redirect()	تحويل العميل إلى URL جديد
Write()	كتابة على الصفحة

الجدول 22. 3. أهم دوال وخصائص الفئة Response

مثال على الكتابة باستخدام Write:

كود — C#
```
Response.Write("<u>This is javascript code</u>");
Response.Write("<script>alert('hiiiii');</script>");
```

كود — VB.NET
```
Response.Write("<u>This is javascript code</u>")
Response.Write("<script>alert('hiiiii');</script>")
```

4. أدوات ASP.net

كما لاحظنا سابقا، فإن تعريف أي اداة من أدوات ASP.net يتم من خلال HTML مكتوب في aspx.*، وفيما عدا ذلك فهي شديدة الشبه بالأدوات العادية في تطبيقاتنا من ناحية المظهر وتعاملك معها كمبرمج ، إلا انها تختلف في الأداء وما وراء الكود بشكل قطعي.
وكذا الأمر بالنسبة للأحداث ايضاً والتي تم عمل ضغط لها لتتناسب مع عالم الويب ، ومع اي Event يتم نقل البيانات إلى السيرفر مباشرة.

خاصية AutoPostBack

تتيح لك هذه الخاصية النقل إلى السيرفر مباشرة مع اي تحديث فيها ، تجد هذه الخاصية في مربعات النص Text Box وادوات الاختيار والتحديد Check Box & Radio Buttons والقوائم بانواعها List Box و Combo Box، والقيمة الافتراضية لها هي false.

4. 1. الخصائص الأساسية لأدوات الويب

تشتق جميع أدوات الويب من الفئة System.Web.UI.Control والتي نجد لها الخصائص والدوال التالية:

الخاصية أو الدالة	الوصف
Controls	تعيد جميع الأدوات الأبناء لهذه الأداة
HasControls()	تعيد قيمة منطقية بوجود أو عدم وجود أدوات داخل هذه الأداة
ID	الاسم الموحد لكل اداة - لا يمكن تكراره-
Page	تعود بمتغير على الصفحة التي تحتوي هذه الاداة
Parent	الأداة الحاضنة لهذه الأداة
Visible	ظهور او اختفاء هذه الاداة

الجدول 22 .4. أهم دوال وخصائص الفئة Control

كما تقدم الفئة System.Web.UI.WebControls.WebControl للأدوات المشتقة منها بعض الخصائص المتعلقة بالمظهر والعرض، منها:

الخاصية	الوصف
BackColor	لون الخلفية
BorderColor	لون الحدود
BorderStyle	ستايل الحدود
BorderWidth	عرض الحدود
Enabled	تفعيل أو عدم تفعيل الأداة
CssClass	ال class الخاص بال styles لهذه الأداة
Font	معلومات الخط من الحجم والاسم وخلافه لهذه الاداة
ForeColor	لون خط الكتابة
Height	الطول
Width	العرض
TabIndex	موقعها من التنقل باستخدامTab
ToolTip	في حالة وجود Tips للاداة

الجدول 22. 5. أهم دوال وخصائص الفئة WebControl

مثال : استعراض اسماء جميع الأدوات في الفورم

باستخدام ال Collection الناتج عن الخاصية Controls اضافة للدالة HasControls() لمعرفة وجود أدوات من عدمه ، يمكنك كتابة الكود التالي:

كود C#

```
string Information = "";
if (myPanel.HasControls())
{
    foreach (Control c in PanelName.Controls)
    {
        if (!object.ReferenceEquals(c.GetType(),
typeof(System.Web.UI.LiteralControl)))
        {
            Information += string.Format("Control Name: {0} <br/>",
c.ToString());
            Information += string.Format("ID: {0} <br/>", c.ID);
            Information += string.Format("Control Visible: {0} <br/>",
c.Visible);
            Information += string.Format("ViewState: {0} <br/>",
c.EnableViewState);
            Information += "<br/><hr/><br/>";
            Response.Wite(Information);
        }
    }
}
```

كود VB.NET

```
Dim Information As String = ""
If myPanel.HasControls() Then
    For Each c As Control In PanelName.Controls
        If Not Object.ReferenceEquals(c.[GetType](),
GetType(System.Web.UI.LiteralControl)) Then
            Information += String.Format("Control Name: {0} <br/>",
c.ToString())
            Information += String.Format("ID: {0} <br/>", c.ID)
            Information += String.Format("Control Visible: {0} <br/>",
c.Visible)
            Information += String.Format("ViewState: {0} <br/>",
c.EnableViewState)
            Information += "<br/><hr/><br/>"
            Response.Wite(Information)

        End If
    Next
End If
```

مثال : انشاء أدوات وقت التصميم

كود	C#

```
TextBox t1 = new TextBox();
t1.ID = string.Format("dynamict1");
myPanel.Controls.Add(t1);
```

كود	VB.NET

```
Dim t1 As New TextBox()
t1.ID = String.Format("dynamict1")
myPanel.Controls.Add(t1)
```

أقسام الأدوات في ASP.net

تنقسم الأدوات الموجودة في ASP.net لعدة أنواع رئيسية:

Simple controls

الأدوات التي تتبع ASP.net ولكنها من عناصر HTML اساساً مثل مربعات النص TextBox والعنوان Label وأزرار الأمر Buttons وخلافه.

Rich controls

مجموعة من أدوات ASP.net العادية ولكنها اكثر تشعباً وخصائص ، من امثلتها اداة التقويم Calendar واداة عرض الشجرة TreeView والقوائم Menu وخلافه

Data controls

هي ادوات تعتمد على الربط بقاعدة البيانات ، من اشهر امثلتها. GridView

Validation controls

ادوات التحقق ، هي ادوات سيرفر ولكن يتم تنفيذها عند العميل، حيث يتم تطبيق محتويات JavaScript فيها لتنفيذ بعض عمليات التحقق.

Login Controls

مجموعة من الادوات المتكاملة لتسجيل الدخول وخلافه.

Web part controls

مجموعة ادوات مخصصة لاتاحة الفرصة لمستخدم موقعك للتحكم في لون وخصائص اجزاء الصفحة.

هناك بالطبع HTML Controls والتي تحدثنا عنها سابقا.

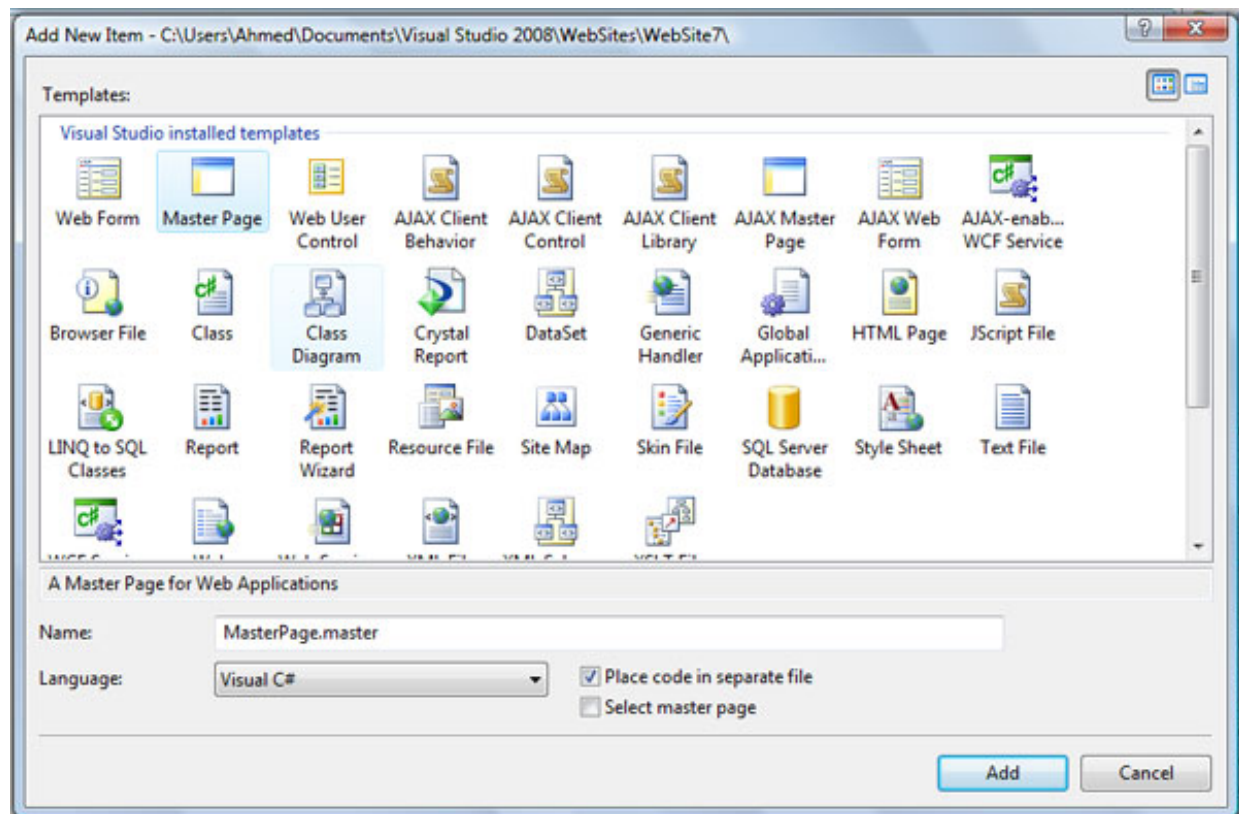
سنحاول ضمن الدروس القادمة التعرف على بعض هذه الأدوات الخاصة التي ستمر علينا للمرة الأولى ما عدا ال GridView والتي سنؤجل الحديث عنها حتى قسم ال ASP.net + قواعد البيانات.

5. Master Pages

لعلك تلاحظ في عدد كبير من المواقع ان جزءاً ثابتاً من الموقع لا يتم تغييره في كل الصفحات، حيث يبدو كجزء من الصفحة ويتفاعل معها ولكنه يبدو مكرراً في كل الصفحات.

في الواقع هذا الجزء ليس موجوداً في كل الصفحات ، لتتخيل ان الموقع به 100 صفحة وقمت بعمل نفس الجزء في كل مرة ، ثم رغبت في تغيير احد خصائصه ، فستضطر للتعديل في ال 100 صفحة كاملة.

من هنا ظهر مبدأ ال include في ASP والذي كان يتيح لك عمل Include لصفحة بعينها في اي جزء من صفحتك ، ومع ASP.net ظهرت لنا الMaster Pages والتي تجعل من صفحة ما صفحة رئيسية لكل الصفحات.

سنجرب هذا المبدأ سوية ، قم باضافة New Item واختر Master Page بالشكل التالي:

الصورة 22. 8. اضافة MasterPage للمشروع.

بعد انشاءك لها ، اول ملا تلاحظه في صفحة الكود هو وجود العنصر الجديد التالي:

كود	ASP.NET

```
<asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server">

</asp:ContentPlaceHolder>
```

يمثل هذين الوسمين الصفحة التي سيتم عمل Master Page لها ، قم بوضع اكوادك المختلفة قبل وبعد هذه المنطقة ، اما هذه المنطقة فسيتم عرض الصفحة الرئيسية فيها.

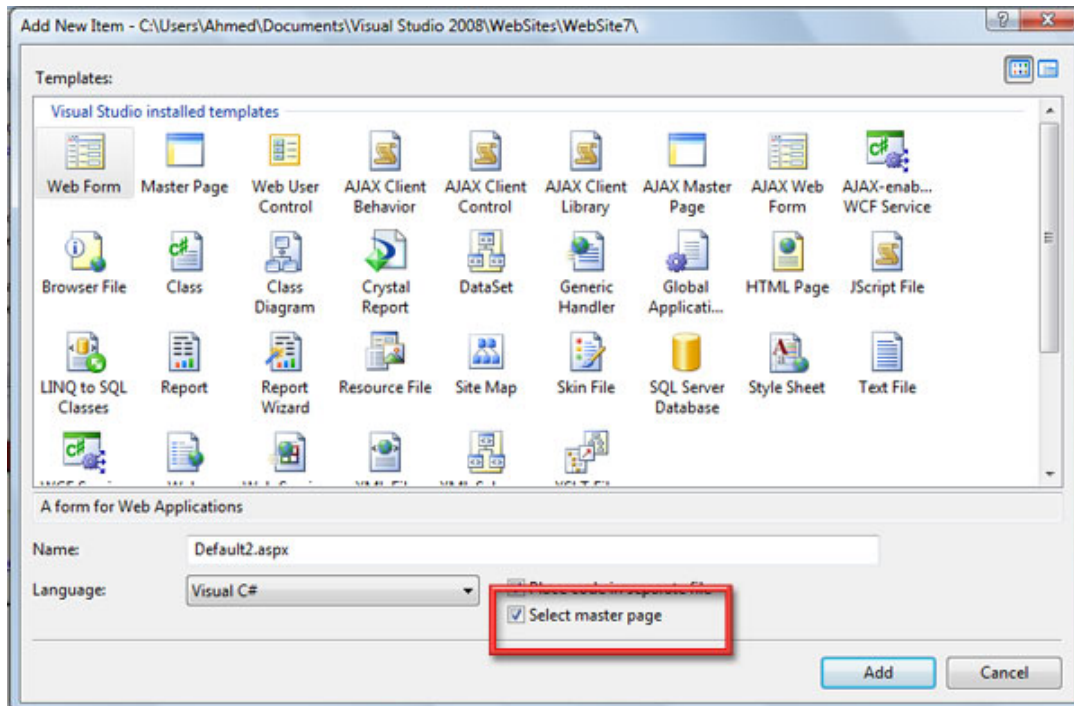
ليس هذا فقط ، اي أن كتابة الكود فوق وتحت الوسم لا يعني انها ستظل فوق وتحت الصفحة فقط ، في المثال التالي سنجعل الصفحة الرئيسية تظهر في مربع صغير فقط وسط الصفحة فيما ال 9 مربعات الاخرى تحتوي على بيانات ما ، بالصورة التالية مثلاً:

الصورة 22. 9. صفحة ال MasterPage.

الكود الخاص بهذه الصفحة سيكون بالشكل التالي:

كود ASP.NET

```
<%@ Master Language="C#" AutoEventWireup="true" CodeFile="MasterPage.master.cs"
Inherits="MasterPage" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>Untitled Page</title>
    <asp:ContentPlaceHolder id="head" runat="server">
    </asp:ContentPlaceHolder>
</head>
<body>
    <form id="form2" runat="server">
    <div>
    <center>
    <table border="1" width=80%>
       <tr>
            <td>hiiii, i am here</td>
            <td><img src="http://vb4arab.com/vb/attachments/logo.jpg" /></td>
            <td>any thing</td>
       </tr>
       <tr>
            <td>hii again</td>
            <td>
        <asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server">
        </asp:ContentPlaceHolder>
        </td>
        <td><input type="button" /></td>
    </tr>
    <tr>
    <td colspan=2>large TD :)</td>
    <td>bye</td>
    </tr>
    </table>
    </center>
    </div>
    </form>
</body>
</html>
```

وطبعاً يمكن تصميمها من خلال ال Designer وليس من الكود فقط ، بعد الانتهاء من التصميم قم بانشاء صفحة جديدة ، وفي شاشة طلب Master Page قم بتحديدها له بالشكل التالي:

الصورة 22. 10. اعداد الصفحة لتكون تلقائيا MasterPage.

لاحقاً سيطلب منك تحديدها في شاشة اخرى ، والآن مع اي صفحة ترغب في حصولها على نفس الشكل ستقوم باعطاءها نفس القيمة ، حتى لو كانت صفحة قديمة يمكنك تعديل الخاصية التالي في Page:

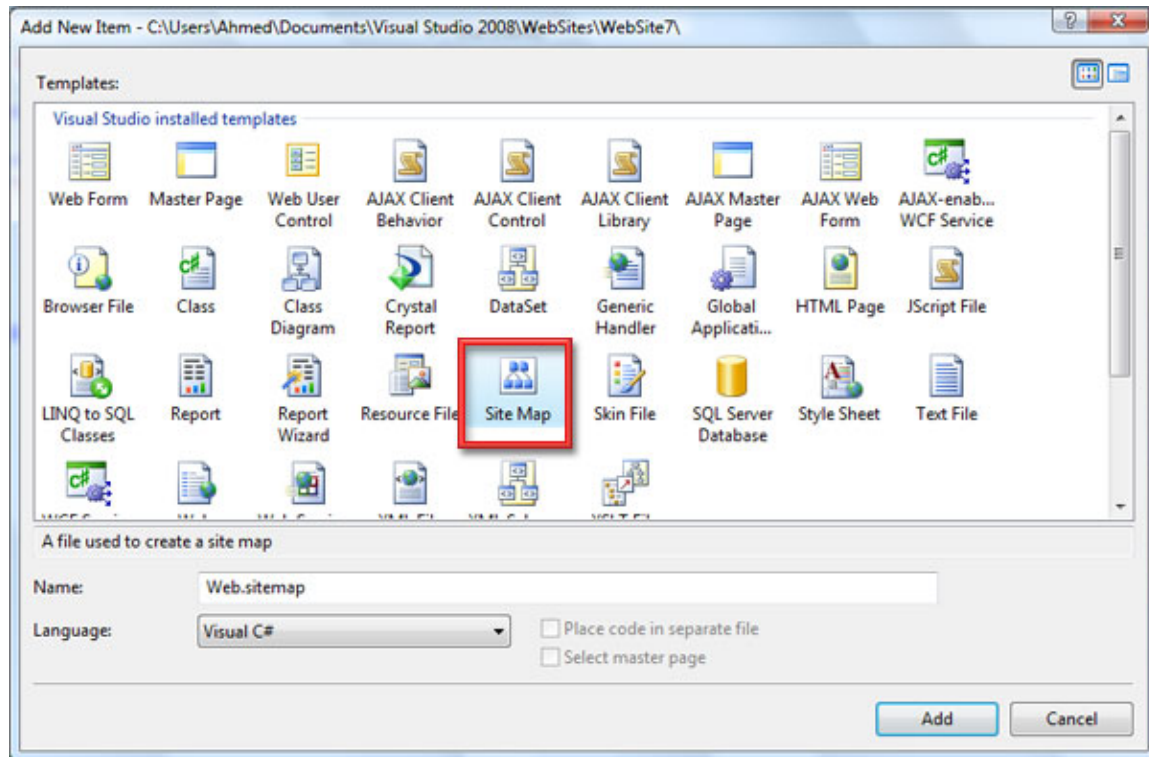
ومراعاة وجود هذا الوسم تحديداً:

كود ASP.NET

```
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1"
Runat="Server"></asp:Content>
```

لكن لا تنس ان لا تحتوي على تكرارات مثل ال head ووسم html .

6. التعامل مع Sitemap

من خلال تعاملك مع كائن ال SiteMap والتي تتيح لك تعريف محتويات موقعك وترتيبها ، تستطيع لاحقاً تشكيل قوائم وعرض شجري لصفحات موقعك، لنبدء اولاً باضافة sitemap بالشكل التالي:

الصورة 22. 11. اضافة SiteMap.

ستجد الكود التالي افتراضياً:

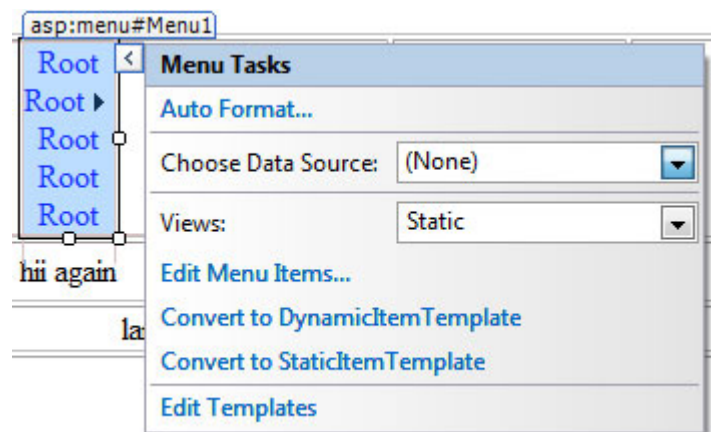
XML | كود

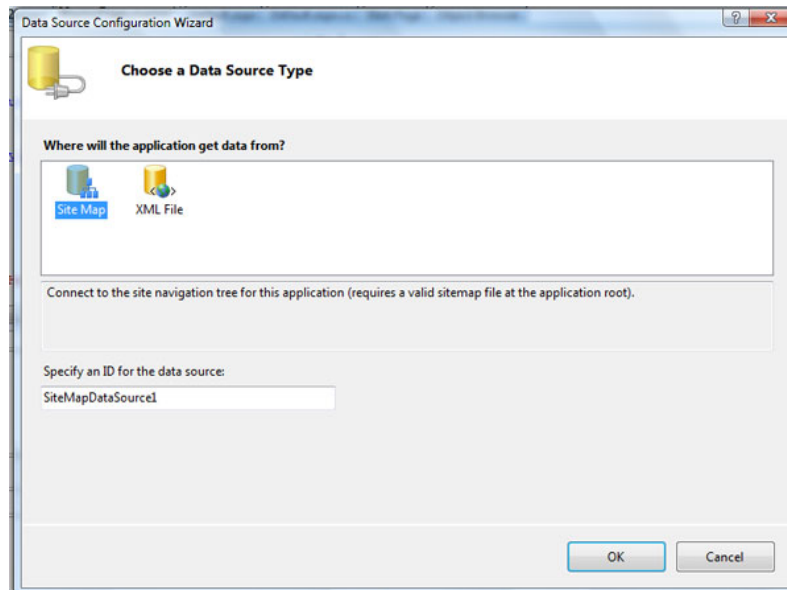
```xml
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
    <siteMapNode url="" title=""  description="">
        <siteMapNode url="" title=""  description="" />
        <siteMapNode url="" title=""  description="" />
    </siteMapNode>
</siteMap>
```

الآن مهمتنا لتعريف هذه ال Nodes ، لنفترض مثلاً صفحة البداية وتحتها صفحتين مختلفتين:

كود	XML

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
    <siteMapNode url="Default1.aspx" title="صفحة البداية"  description="">
        <siteMapNode url="Default2.aspx" title="صفحة لوجو الويهو !"
description="" />
        <siteMapNode url="Default3.aspx" title="صفحة لوجو المنتدى"
description="" />
    </siteMapNode>
</siteMap>
```

الآن سنحاول الاستفادة من هذه المعلومات ، سننتقل إلى ال MasterPage الخاصة بنا وسنقوم باضافة Menu وتحديد ال DataSource لها ليكون هذا الsitemap :

الصورة 22. 12. تحديد مصدر البيانات.

في الخطوة التالية سيطلب منك تحديد نوع ال DataSource ، اختر Site map

الصورة 22. 13. اختيار ال SiteMap كمصدر للبيانات.

الآن ، جرب تشغيل موقعك والذي سيكون بالشكل التالي:

SiteMapPath

أداة اخرى تتبع نفس المجموعة ، مهمتها تحديد المكان لك بالشكل التالي مثلاً:

Root Node > Parent Node > Current Node

قم بوضعها في ال masterpage ، وستعمل مع كامل صفحاتك بصورة طبيعية

7. أدوات التحقق Validation Controls

هي مجموعة من ادوات التحقق من المدخلات يعمل اغلبها جهة العميل Client Side، وتحتوي على الأدوات التالية:

- CompareValidator

ارنة المدخلات في جهة بمدخلات أخرى ، تفيد مثلاً في حالة اعادة تأكيد كلمة المرور ، ربط هذه الأداة كافي لتطبيق هذا التحقق

- RangeValidator

أكد من ان المدخلات تقع ضمن نطاق معين يتم تحديده

- RequiredFieldValidator

للتأكد من أن المستخدم قام بادخال بيانات.

- RegularExpressionValidator

يمكنك تحديد نوع من التحقق بناء على Reqular Expression تفيدك مثلاً في حالة التحقق من صحة موقع أو بريد الكتروني او رقم هاتف ، لمعرفة المزيد عن ال Reqular Expressions يمكنك مراجعة هذا الدرس - عربي - :

رابط

http://www.arabteam2000-forum.com/index.php?showtopic=77787

ايضاً يمكنك الاستفادة من هذين الموقعين لاستخراج اي RegularExpression ترغب به:

رابط

http://regexlib.com/DisplayPatterns.aspx
http://www.regular-expressions.info/

- CustomValidator

تتيح لك بناء اجراءات التحقق الخاصة بك.

- ValidationSummary

تعرض موجز عمليات التحقق التي تمت في هذا الفورم.

تحتوي هذه المجموعة على الخصائص التالية:

ControlToValidate : أداة الادخال المطلوب التحقق منها.

Display : العرض في حالة حدوث الخطأ.

ErrorMessage : رسالة الخطأ.

ForeColor : لون رسالة الخطأ.

الآن لتجربة هذه المجموعة ، سنقوم بتصميم فورم تسجيل بيانات بسيطة ، تحتوي على الاسم ، ورقم المرور وتأكيد كلمة المرور ، والبريد الالكتروني والعمر .

-الاسم لا يمكن ان يكون خالياً : RequiredFieldValidator

-كلمة المرور وتأكيدها لا بد ان يكونوا منطبقين: CompareValidator

-الايميل لا بد ان يكون صحيحاً : RegularExpressionValidator

-العمر لا بد أن يكون بين 10 و 50 : RangeValidator

-وفي النهاية سنعرض نتائج التحقق جميعها...

قم بتصميم الفورم ، وضع كل اداة بجانب الحقل الخاص بها ليكون الشكل التالي:

Name:

you must eneter Name

Age:

Age must be between 10 and 50

Password:

Retype Password:

The passwords must be the same

E-mail:

invalid email address

Go On

Summery:

- Error message 1.
- Error message 2.

الصورة 22. 14. محتويات الصفحة و عرض الأخطاء.

والكود:

كود	ASP.NET

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default4.aspx.cs"
Inherits="Default4" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>  Name:<br />
 <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
        <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"
            ControlToValidate="TextBox1" ErrorMessage="you must eneter
Name"></asp:RequiredFieldValidator><br /><br />
        Age:<br />
        <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
        <asp:RangeValidator ID="RangeValidator1" runat="server"
            ControlToValidate="TextBox2" ErrorMessage="Age must be between 10
and 50"
            MaximumValue="50" MinimumValue="10"></asp:RangeValidator>
        <br /><br />
        Password:<br />
        <asp:TextBox ID="TextBox3" runat="server"></asp:TextBox>
        <br /><br />
        Retype Password:<br />
        <asp:TextBox ID="TextBox4" runat="server"></asp:TextBox>
        <asp:CompareValidator ID="CompareValidator1" runat="server"
            ControlToCompare="TextBox3" ControlToValidate="TextBox4"
            ErrorMessage="The passwords must be the
same"></asp:CompareValidator><br /><br />
        E-mail:<br />
 <asp:TextBox ID="TextBox5" runat="server"></asp:TextBox>
        <asp:RegularExpressionValidator ID="RegularExpressionValidator1"
runat="server"
            ControlToValidate="TextBox5" ErrorMessage="invalid email address"
            ValidationExpression="^((?&gt;[a-zA-Z\d!#$%&amp;'*+\-
/=?^_`{|}~]+\x20*|&quot;((?=[\x01-\x7f])[^&quot;\\]|\\[\x01-
\x7f])*&quot;\x20*)*(?&lt;angle&gt;&lt;))?((?!\.)(?&gt;\.?[a-zA-
Z\d!#$%&amp;'*+\-/=?^_`{|}~]+)+|&quot;((?=[\x01-\x7f])[^&quot;\\]|\\[\x01-
\x7f])*&quot;)@(((?!-)[a-zA-Z\d\-]+(?&lt;!-)\.)+[a-zA-
Z]{2,}|\[(((?(?&lt;!\[)\.)(25[0-5]|2[0-4]\d|[01]?\d?\d)){4}|[a-zA-Z\d\-]*[a-zA-
Z\d]:((?=[\x01-\x7f])[^\\\[\]]|\\[\x01-
\x7f])+)\])(?(angle)&gt;)$"></asp:RegularExpressionValidator>
        <br /><br />
        <asp:Button ID="Button1" runat="server" Text="Go On" /><br /><br />
        <Summery:r />
        <asp:ValidationSummary ID="ValidationSummary1" runat="server" />
    </div>
    </form>
</body>
</html>
```

طبعاً قمت بتغيير ErrorMessage لكل منهم ، وقمت بتحديد ال ControlToValidate لكل واحدة فيهم.

ال Regular Expression اللازم لعملية الايميل لو لم تكن قد راجعت بعض الدروس السابقة هو:

```
"^((?>[a-zA-Z\d!#$%&'*+\-/=?^_`{|}~]+\x20*|"((?=[\x01-\x7f])[^"\\]|\\[\x01-
\x7f])*"\x20*)*(?<angle><))?((?!\.)(?>\.?[a-zA-Z\d!#$%&'*+\-
/=?^_`{|}~]+)+|"((?=[\x01-\x7f])[^"\\]|\\[\x01-\x7f])*")@(((?!-)[a-zA-Z\d\-
]+(?<!-)\.)+[a-zA-Z]{2,}|\[(((?(?<!\[)\.)(25[0-5]|2[0-4]\d|[01]?\d?\d)){4}|[a-zA-
Z\d\-]*[a-zA-Z\d]:((?=[\x01-\x7f])[^\\\[\]]|\\[\x01-\x7f])+)\])(?(angle)>)$"
```

وبالنسبة للمقارنة ، فهناك خاصية ControlToCompare اضافة لخاصية ControlToValidate، وفي ال Range خصائص MinimumValue و MaximumValue.

والآن جرب ارتكاب الاخطاء خطأ وراء الآخر ، وشاهد النتيجة:

Name:
you must eneter Name

Age:
100

Password:
12

Retype Password:
1 The passwords must be the same

E-mail:
A@a. invalid email address

Go On

Summery:r />

- you must eneter Name
- The passwords must be the same
- invalid email address

الصورة 22. 15. عرض الأخطاء في الأداة Summery .

نقاط سريعة

- يمكنك معرفة المزيد عن Web Parts من هنا:

رابط

http://msdn.microsoft.com/en-us/library/e0s9t4ck.aspx

- كما ترى فالتصاميم التي نقوم بها حتى اللحظة بدائية جداً ، هناك طريقين لتحسين تصاميمك، الحل القديم يعتمد على استخدام لل CSS ، وهذه يمكنك معرفة المزيد عنها هنا:

رابط

http://www.w3.org/Style/CSS/learning

الحل الجديد ابتداء من ASP.net هو استخدام ما يعرف باسم Themes، يمكنك البدء فيها من هنا

رابط

http://msdn.microsoft.com/en-us/library/ykzx33wh.aspx

8. State Management

لعلك ومن خلال تجاربك في تطوير Desktop Application تدرك انك وقت تعريفك لمتغير ما فسيظل هذا المتغير محتفظاً بقيمته حتى انهاء البرنامج أو الخروج خارج ال Scope الخاص بهذا المتغير ، إلا أن هذا الوضع مختلف تماماً في ASP.net حيث ستضيع قيم المتغيرات مع أول تعديل ، لذا كان من اللازم عليك أن تقوم بتخزين متغيراتك وقيم بحيث يمكنك استرجاعها ، وهو ما سنتعرف على بعض تقنياته في هذا الدرس.

يمكنك تخزين القيم بواحدة من ستة طرق اساسية:

- View state .
- Control state .
- Application-Level Variable .
- Cache .
- Session .
- Cookies .

8. 1. Control State

هي الطريقة الأسهل للاحتفاظ بالبيانات، كل ما عليك هو تغيير القيمة EnableViewState لأي أداة حتى ولو للصفحة ككل ، في هذه الحالة القيم الموجودة في أي من أدواتك لن تتأثر وستظل محتفظة بقيمتها حتى اغلاق الصفحة.

لو فتحت ال HTML الناتج عن المتصفح لصفحة تستخدم View State ستجد الجزء التالي حيث يحتفظ ال ViewState بقيمته.

كود — ASP.NET
```
<input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE"
value="/wEPDwUKMTIxNDIyOTM0Mg9kFgICAw9kFgICAQ8PFgIeBFRleHQFBWFobWVkZGRkCq8FGqyXB
P0pBDpvdnycvM5zSQE=" />
```

خطأ شائع

أحياناً ما تحتاج لإن تقوم بملء محتويات قائمة لديك في حدث ال Form_Load، اغلب المطورين تستخدم الاسلوب التالي:

كود — C#
```
ListBox1.Items.Add("Ahmed");
ListBox1.Items.Add("Mohammed");
ListBox1.Items.Add("Ali");
```

كود — VB.NET
```
ListBox1.Items.Add("Ahmed")
ListBox1.Items.Add("Mohammed")
ListBox1.Items.Add("Ali")
```

في الواقع هذه الطريقة متعبة جداً، خصوصاً لو كان ملء القائمة يتم من خلال قاعدة البيانات!!! الحل البديل، هو الاعتماد على خاصية EnableViewState، واستخدام الخاصية IsPostBack للتأكد من أننا نقوم بملأها في المرة الأولى فقط بالشكل التالي:

كود — C#
```
if (!IsPostBack)
{
    ListBox1.Items.Add("Ahmed");
    ListBox1.Items.Add("Mohammed");
    ListBox1.Items.Add("Ali");
}
```

كود — VB.NET
```
If Not IsPostBack Then
    ListBox1.Items.Add("Ahmed")
    ListBox1.Items.Add("Mohammed")
    ListBox1.Items.Add("Ali")
End If
```

8. 2. ViewState

طريقة أخرى تعتمد على نفس المفهوم ولكن بعيداً عن الأدوات، حيث يمكنك مثلاً كتابة الكود التالي لتخزين قيمة ما بنفس الطريقة:

كود C#
```
ViewState["CustomViewStateItem"] = "Ahmed";
```

كود VB.NET
```
ViewState("CustomViewStateItem") = "Ahmed"
```

ولاستعادته:

كود C#
```
Label1.Text = (string)ViewState["CustomViewStateItem"];
```

كود VB.NET
```
Label1.Text = DirectCast(ViewState("CustomViewStateItem"), String)
```

8. 3. Session

طريقة سهلة التعامل جداً، وبنفس طريقة ViewState السابقة مع اختلاف التكنيك فقط، للكتابة:

كود C#
```
Session["mySession"] = "Ahmed";
```

كود VB.NET
```
Session("mySession") = "Ahmed"
```

والاستعادة:

كود C#
```
string name = (string)Session["mySession"];
```

كود VB.NET
```
Dim name As String = DirectCast(Session("mySession"), String)
```

يمكنك عمل Remove لأي قيمة مخزنة في الكائن Session بالشكل التالي:

كود C#
```
Session.Remove["mySession"];
```

كود VB.NET
```
Session.Remove("mySession")
```

كما أن هناك خاصية Timeout والتي تحدد فترة احتفاظ ال Session بقيمته بالدقائق ، وافتراضياً هي 20 دقيقة:

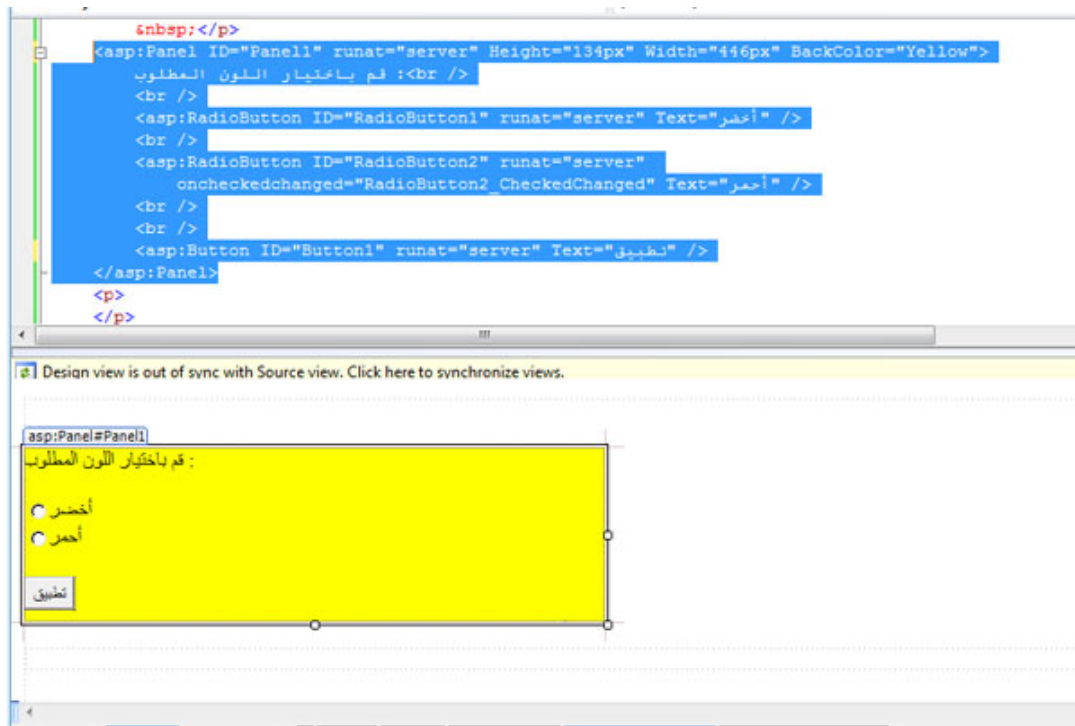
كود C#
```
Session.Timeout = 5;
```

كود VB.NET
```
Session.Timeout = 5
```

8. 4. Cookies

طريقة أخرى من وسائل الاحتفاظ بالقيم ولكن عند جهاز العميل، حيث يتم الاحتفاظ بهذه القيم في ملفات نصية (في اغلب الاحيان) ولكن ينبغي مراعاة ان يكون متصفح العميل يسمح بمثل هذه العملية ، وهي الطريقة الأكثر شهرة في الاحتفاظ بمعلومات الدخول والتسجيل في المواقع المختلفة وهي مشتقة من الفئة System.Web.HttpCookie

سنجرب الآن مثالاً على الكوكيز ، حيث سنقوم بتسجيل اختيار المستخدم من الألوان - اللون الاخضر أو الأحمر مثلاً -، ومع كل تشغيل للموقع في حالة وجود كوكيز يتم استخدام اللون المفضل ، وإلا يتم اللجوء للون الافتراضي وهو الأصفر مثلاً.

سنقوم برسم الصفحة اولاً باللون الافتراضي الأصفر:

الصورة 22 .16. المظهر العام للصفحة.

الكود الخاص بها:

كود	ASP.NET

```
<asp:Panel ID="Panel1" runat="server" Height="134px" Width="446px"
BackColor="Yellow">
    قم باختيار اللون المطلوب :<br />
    <br />
    <asp:RadioButton ID="RadioButton1" runat="server" Text="أخضر"
GroupName="colors" />
    <br />
    <asp:RadioButton ID="RadioButton2" runat="server"
        oncheckedchanged="RadioButton2_CheckedChanged" Text="أحمر"
GroupName="colors"/>
    <br />
    <br />
    <asp:Button ID="Button1" runat="server" Text="تطبيق"
onclick="Button1_Click1" />
</asp:Panel>
```

والآن ، سنقوم بانشاء كوكيز نخزن فيه اللون المختار حالياً بالشكل التالي في حدث الضغط على زر (تطبيق) ، وسنعطي هذا الكوكيز تاريخاً للانتهاء:

C#	كود

```
string color;
if (RadioButton1.Checked)
{
    color = "Green";
    Panel1.BackColor = System.Drawing.Color.Green;
}
else if (RadioButton2.Checked)
{
    color = "Red";
    Panel1.BackColor = System.Drawing.Color.Red;
}
else
    color = "";
HttpCookie Cookie = new HttpCookie("myColor", color);
Cookie.Expires = DateTime.Parse("01/01/2009");
Response.Cookies.Add(Cookie);
```

VB.NET	كود

```
Dim color As String
If RadioButton1.Checked Then
    color = "Green"
    Panel1.BackColor = System.Drawing.Color.Green
ElseIf RadioButton2.Checked Then
    color = "Red"
    Panel1.BackColor = System.Drawing.Color.Red
Else
    color = ""
End If
Dim Cookie As New HttpCookie("myColor", color)
Cookie.Expires = DateTime.Parse("01/01/2009")
Response.Cookies.Add(Cookie)
```

والآن في حدث ال Form_Load للتشغيل، سنقوم بقراءة الكوكيز، وفي حالة وجود myColor فسيتم معرفة قيمته:

C#	كود

```
if (Request.Cookies["myColor"] != null)
{
    string color = Request.Cookies["myColor"].Value;
    if (color == "Green")
        Panel1.BackColor = System.Drawing.Color.Green;
    else if (color == "Red")
        Panel1.BackColor = System.Drawing.Color.Red;
}
```

كود	VB.NET

```
If Request.Cookies("myColor") IsNot Nothing Then
Dim color As String = Request.Cookies("myColor").Value
    If color = "Green" Then
        Panel1.BackColor = System.Drawing.Color.Green
ElseIf color = "Red" Then
        Panel1.BackColor = System.Drawing.Color.Red
    End If
End If
```

8 .5. Application

طريقة أخرى مشتقة من HttpApplicationState، ولكن من خلالها لا يتم تخزين البيانات لعميل واحد ، بل يتم تخزين بيانات يمكن ان تصل لجميع العملاء ، ابسط مثال لها هو عدد الزوار والذي ينبغي ان يكون مؤثراً عند جميع العملاء وليس عميل واحد فقط.

تحتوي هذه الفئة على الدوال والخصائص التالية:

الخاصية أو الدالة	الوصف
Add()	اضافة عنصر جديد باسم جديد للقائمة
AllKeys	استعراض جميع العناصر الموجودة
Count	معرفة عدد العناصر الموجودة
Lock(), Unlock()	السماح او عدم السماح بتعديل المجموعة
RemoveAt() Remove(),RemoveAll()	تستخدم لحذف عنصر في نقطة معينة ، أو باسم معين ، أو حذف الكل

الجدول 22 .6. أهم دوال وخصائص الفئة HttpApplicationState

يمكنك اضافة عنصر جديد Application جديد بالشكل التالي:

كود	C#

```
Application["Visitors"] = 1;
```

كود	VB.NET

```
Application("Visitors") = 1
```

ويمكن استعادتها في مربع عنوان مثلاً:

كود	C#

```
labelVistor.Text = (string)Application["Visitors"];
```

كود	VB.NET

```
labelVistor.Text = DirectCast(Application("Visitors"), String)
```

ولتعديل قيمة مثلاً:

كود	C#

```
Application["Visitor"] = ((int)Application["Visitor"]) + 1;
```

كود	VB.NET

```
Application("Visitor") = CInt(Application("Visitor")) + 1
```

ملاحظة

لاتنس ان محتويات ال Application قد لا تكون نص فقط ، بل قد تكون مصفوفة من الفئات

8. 6. Cache

تقنية أخرى مختصة بالتطبيق عند جميع أجهزة العملاء ، ولكن يمكنك تحديد فترة معينة فقط لها ، لانشاءها نستخدم الأمر التالي مثلاً:

كود	C#

```
Context.Cache["myCache"] = "hii, my first cache";
```

كود	VB.NET

```
Context.Cache("myCache") = "hii, my first cache"
```

وللاستعادة:

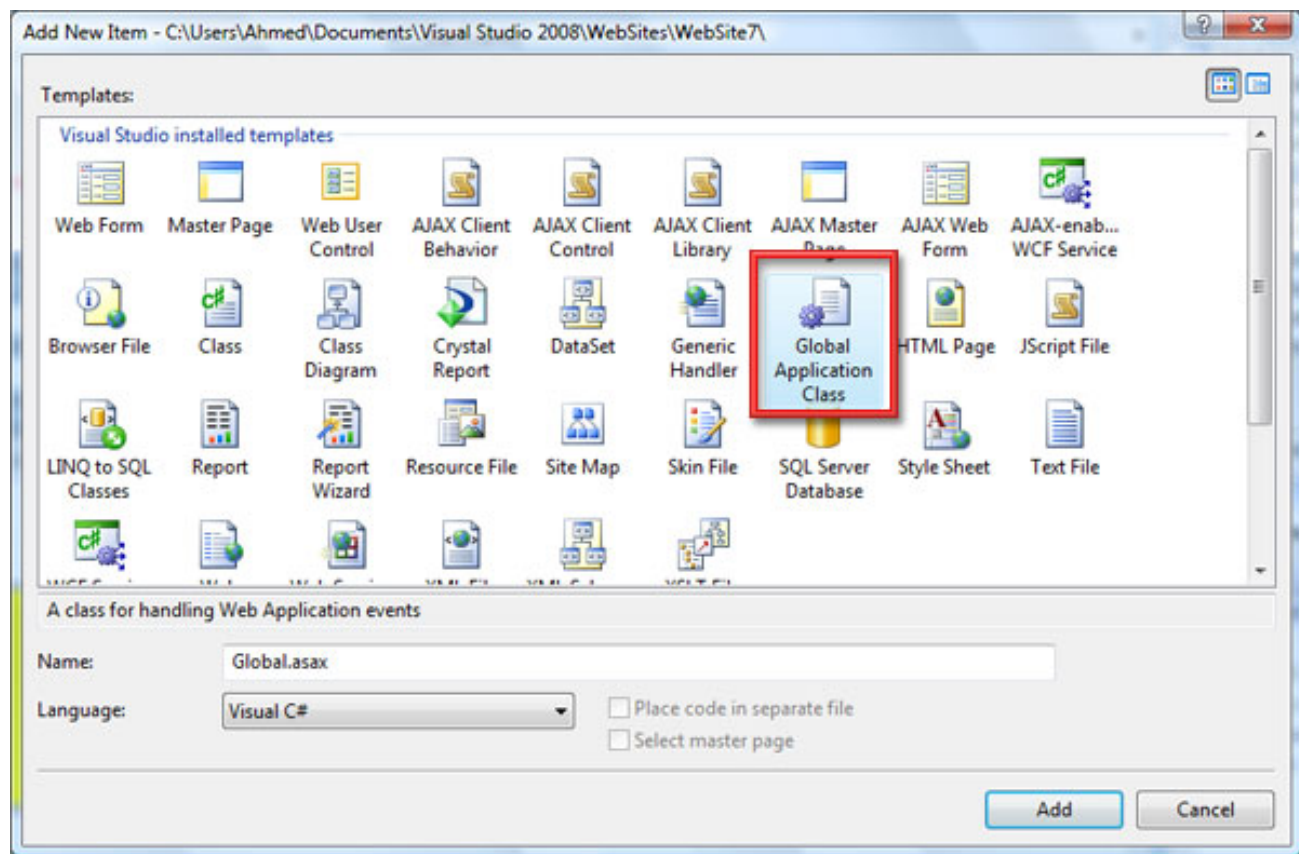
كود　C#

```
string myCache = (string)Context.Cache["myCache"];
```

كود　VB.NET

```
Dim myCache As String = DirectCast(Context.Cache("myCache"), String)
```

8. 7. Global.asax

آخر جزئية لدينا في موضوع التخزين هي استخدام ملف Global.asax لتخزين متغيرات Global ودوال يمكن الوصول إليها من اي صفحة ، قم باضافة عنصر جديد وقم باختيارها بالشكل التالي:

الصورة 22. 17. اضافة ملف Global.asax.

بعد اضافتك لهذه الصفحة ، ستجد بها افتراضياً الأكواد التالية:

كود	C#

```
using System;
namespace myWebSite
{
    public class Global : System.Web.HttpApplication
    {
        protected void Application_Start(object sender, EventArgs e)
        {

        }

        protected void Application_End(object sender, EventArgs e)
        {

        }

        protected void Application_Error(object sender, EventArgs e)
        {

        }

        protected void Session_Start(object sender, EventArgs e)
        {

        }

        protected void Session_End(object sender, EventArgs e)
        {

        }
    }
}
```

كود	VB.NET

```
<%@ Application Language="VB" %>
<script runat="server">
    Sub Application_Start(ByVal sender As Object, ByVal e As EventArgs)
        ' Code that runs on application startup
    End Sub
    Sub Application_End(ByVal sender As Object, ByVal e As EventArgs)
        ' Code that runs on application shutdown
    End Sub
    Sub Application_Error(ByVal sender As Object, ByVal e As EventArgs)
        ' Code that runs when an unhandled error occurs
    End Sub
    Sub Session_Start(ByVal sender As Object, ByVal e As EventArgs)
        ' Code that runs when a new session is started
    End Sub
    Sub Session_End(ByVal sender As Object, ByVal e As EventArgs)
        ' Code that runs when a session ends.
        ' Note: The Session_End event is raised only when the sessionstate mode
        ' is set to InProc in the Web.config file. If session mode is set to
StateServer
        ' or SQLServer, the event is not raised.
    End Sub
</script>
```

وكما لاحظت ، تحوي هذه الصفحة على تعريفات لدوال رئيسية خاصة بالتعامل مع كافة انواع ال State Management، اضافة لحالة حدوث الأخطاء Application_Error وغيرها من دوال تفيد في ادارة كامل صفحات المشروع ، ويمكنك الاستفادة منها في جميع استخداماتك لعناصر State Management المختلفة التي تعرفنا عليها في هذا الدرس.

9. ASP.net و قواعد البيانات

سنأخذ الآن جولة سريعة في عالم قواعد البيانات مع ASP.net ، في الواقع لن تحتاج لأكثر من المعلومات التي تعلمتها في دروس ADO.net ، والتي سنطبقها هنا.

لذا سنقوم بداية بعمل موقع بسيط يقوم فقط بقراءة أسماء من قاعدة البيانات وطباعتها للمستخدم ، سنستخدم قاعدة بيانات تحتوي مبدئياً على اسم المنتج وسعره ، فقط هذا هو الكود ليس أكثر و لا اقل:

كود	C#

```
SqlConnection cn = new SqlConnection(@"Data Source=AHMED-PC\SQLEXPRESS;Initial
Catalog=Items;Integrated Security=True;Pooling=False");
cn.Open();
SqlCommand cmd = new SqlCommand("select * from Items", cn);
SqlDataReader dr = cmd.ExecuteReader();
while (dr.Read())
{
    Response.Write("<b><font color=blue> Item Number: </font></b>" +
dr.GetInt64(0).ToString() + "<br/>");
    Response.Write("<b><font color=blue> Item Name: </font></b>" +
dr.GetString(1) + "<br/>");
    Response.Write("<b><font color=blue>Item Price: </font></b>" +
dr.GetInt64(2).ToString() + "<hr/>");
}
```

كود	VB.NET

```
Dim cn As New SqlConnection("Data Source=AHMED-PC\SQLEXPRESS;Initial
Catalog=Items;Integrated Security=True;Pooling=False")
cn.Open()
Dim cmd As New SqlCommand("select * from Items", cn)
Dim dr As SqlDataReader = cmd.ExecuteReader()
While dr.Read()
    Response.Write("<b><font color=blue> Item Number: </font></b>" +
dr.GetInt64(0).ToString() + "<br/>")
    Response.Write("<b><font color=blue> Item Name: </font></b>" +
dr.GetString(1) + "<br/>")
    Response.Write("<b><font color=blue>Item Price: </font></b>" +
dr.GetInt64(2).ToString() + "<hr/>")
End While
```

والناتج:

Item Number: 1
Item Name: Car Games
Item Price: 50

Item Number: 2
Item Name: Orange
Item Price: 10

Item Number: 3
Item Name: Apple
Item Price: 15

Item Number: 4
Item Name: Fish
Item Price: 85

الصورة 22. 18. نتائج تنفيذ الشفرة.

الآن سنجمل عملية العرض قليلاً، سنستخدم بعض الجداول كما سنضيف لقاعدة البيانات حقل رابط الصورة، سنضيف بعض الصور في مجلد images في نفس مسار البرنامج والتي سنعرضها لكل منتج - الكود سيكون بالشكل التالي:

C#	كود

```
SqlConnection cn = new SqlConnection(@"Data Source=AHMED-PC\SQLEXPRESS;Initial
Catalog=Items;Integrated Security=True;Pooling=False");
cn.Open();
SqlCommand cmd = new SqlCommand("select * from Items", cn);
SqlDataReader dr = cmd.ExecuteReader();
Response.Write("<center><table border=1><tr valign=center align=center
bgcolor=#AAAAAA><td><b>Item No.</b></td><td><b>Item Name</b></td><td><b>Item
Price</b></td><td><b>Sample</b></td></tr>");
while (dr.Read())
{
    Response.Write("<tr><td><b>" + dr.GetInt64(0).ToString() + "</b></td>");
    Response.Write("<td>" + dr.GetString(1) + "</td>");
    Response.Write("<td>" + dr.GetInt64(2).ToString() + "</td>");
    Response.Write(@"<td><img src='images/" + dr.GetString(3) +
"'></td></tr>");
}
Response.Write("</table></center>");
```

VB.NET	كود

```
Dim cn As New SqlConnection("Data Source=AHMED-PC\SQLEXPRESS;Initial
Catalog=Items;Integrated Security=True;Pooling=False")
cn.Open()
Dim cmd As New SqlCommand("select * from Items", cn)
Dim dr As SqlDataReader = cmd.ExecuteReader()
Response.Write("<center><table border=1><tr valign=center align=center
bgcolor=#AAAAAA><td><b>Item No.</b></td><td><b>Item Name</b></td><td><b>Item
Price</b></td><td><b>Sample</b></td></tr>")
While dr.Read()

    Response.Write("<tr><td><b>" + dr.GetInt64(0).ToString() + "</b></td>")
    Response.Write("<td>" + dr.GetString(1) + "</td>")
    Response.Write("<td>" + dr.GetInt64(2).ToString() + "</td>")
    Response.Write("<td><img src='images/" + dr.GetString(3) + "'></td></tr>")
End While
Response.Write("</table></center>")
```

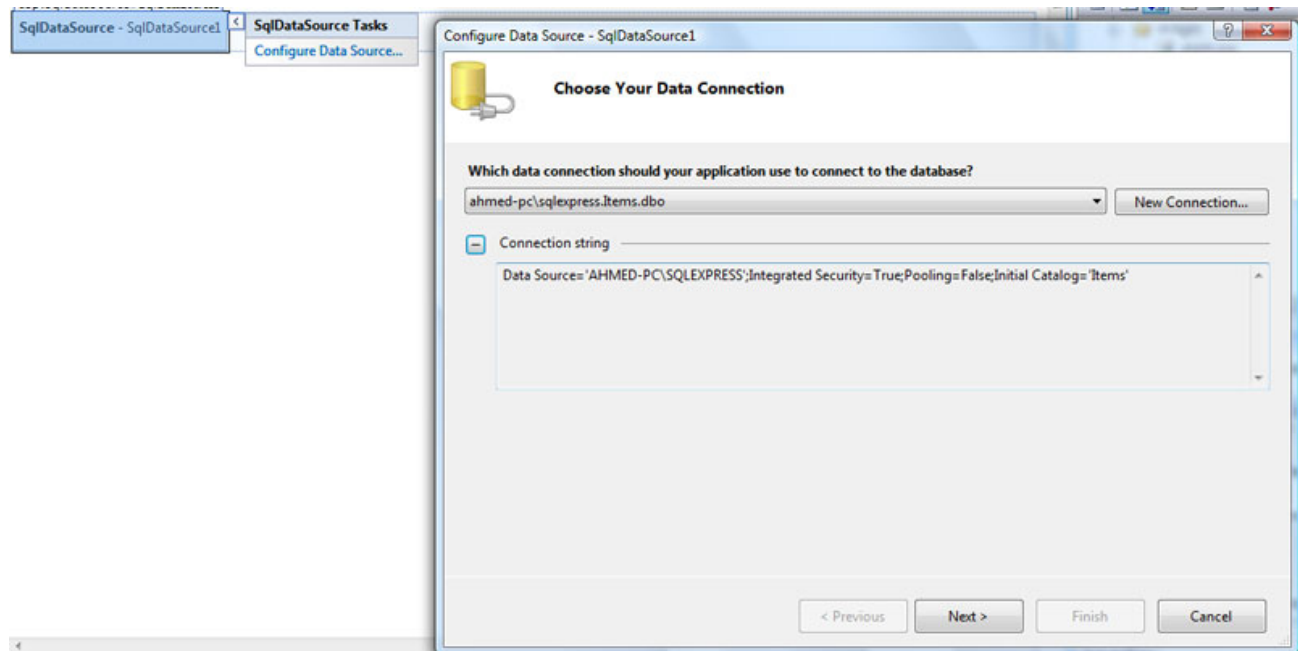
والناتج سيكون بالشكل التالي مثلاً:

Item No.	Item Name	Item Price	Sample
1	Car Games	50	
2	Orange	10	
3	Apple	15	
4	Fish	85	

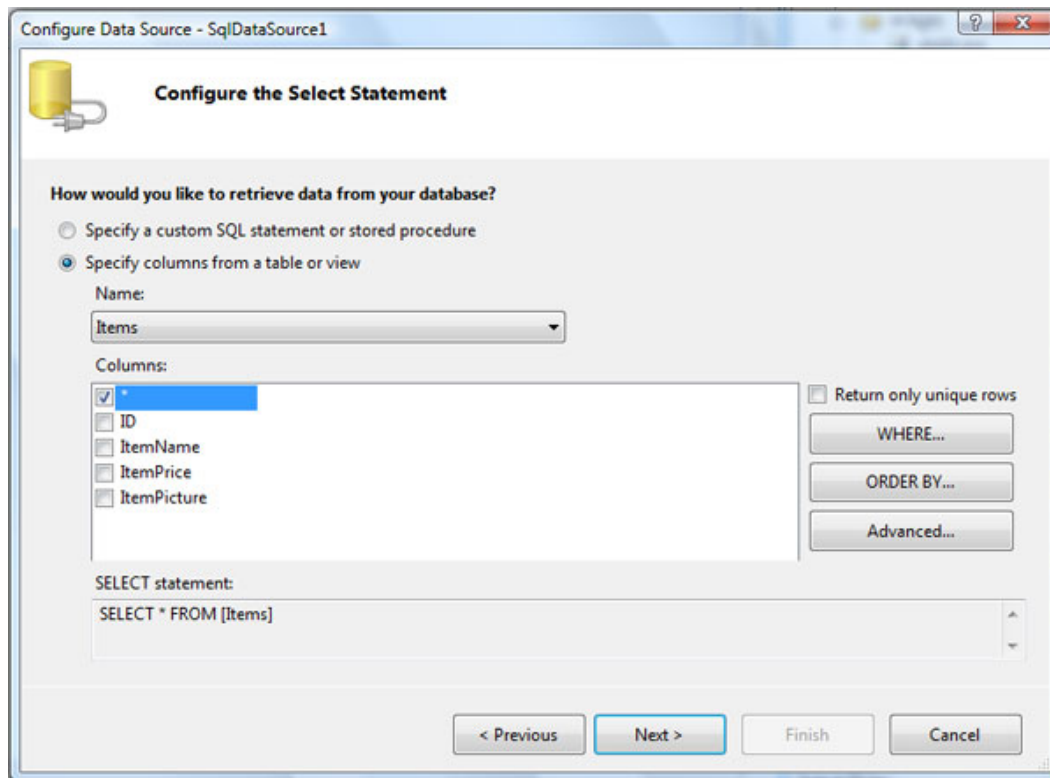
الصورة 22. 19. نتائج تنفيذ الشفرة.

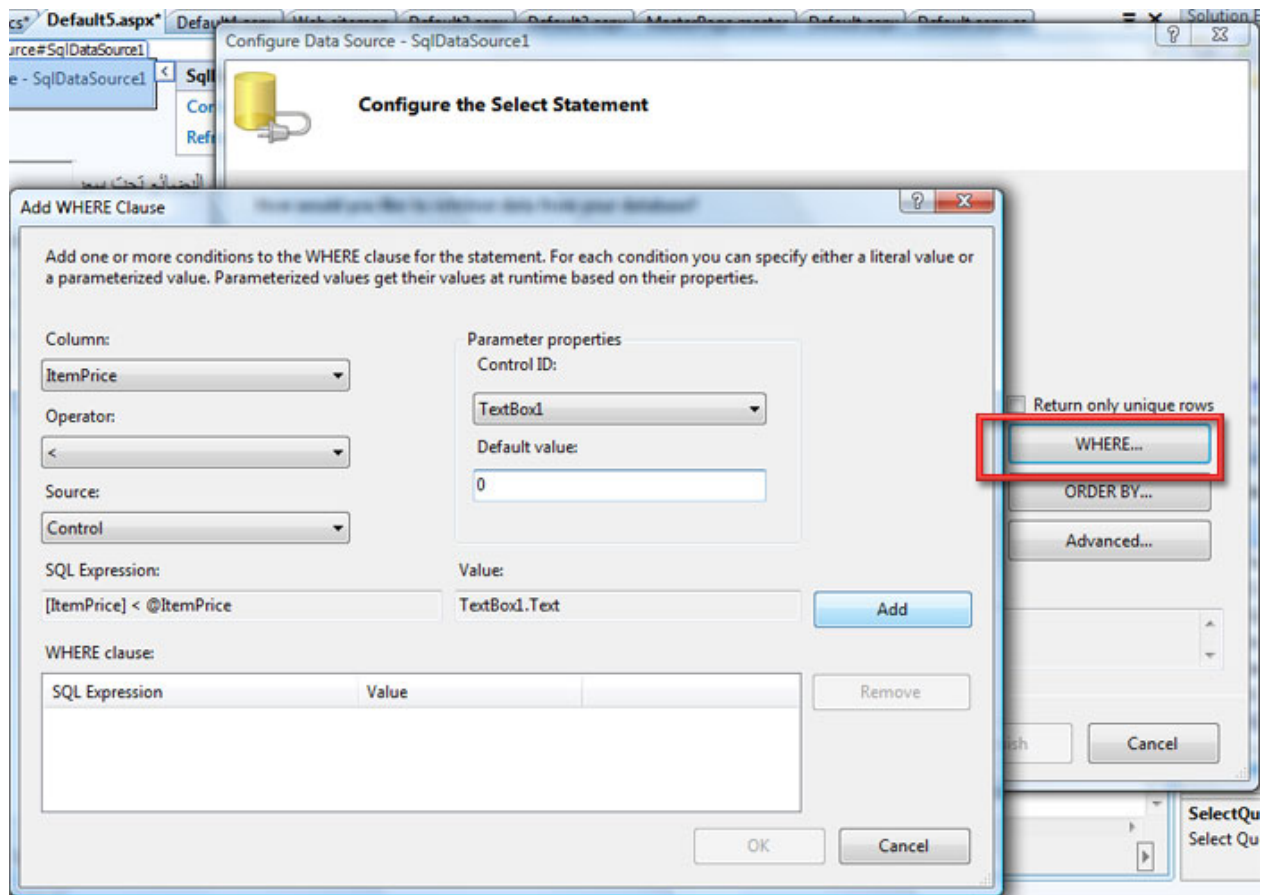
*** هذه هي نفس الطريقة التي كنا نطبقها في عالم ASP Classic ، الآن سنتعرف على بعض التسهيلات التي تتيحها لنا ال net. ...

9. 1. DataGrid

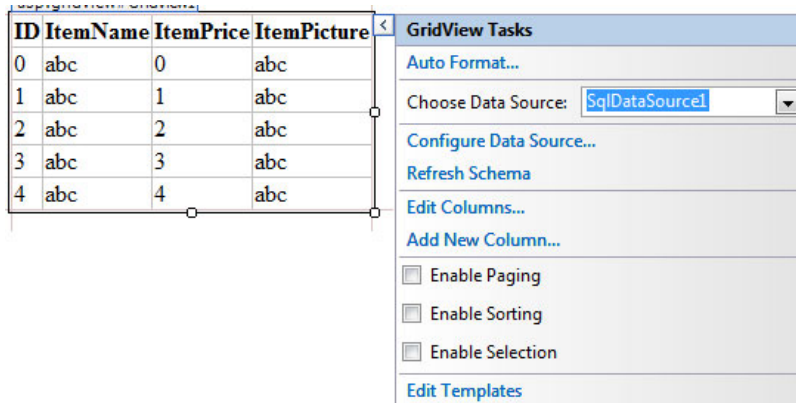
تعتبر هذه الاداة اكثر ادوات البيانات شهرة مع ASP.net، سنجرب الآن التعامل معها...
قم برسم DataGrid، قم برسم مربع نص وزر أمر من اجل عملية البحث، ومن ثم قم بسحب SqlDataSource ونقوم بربطها بجدولنا بالشكل التالي:

الصورة 22. 20. تهيأة مصدر البيانات.

سيطلب منك لاحقاً حفظ ال ConnectionString ، اضغط Next.

في الخطوة الثالثة سيطلب منك تحديد مصدر البيانات ، قم باختيار كافة محتويات الحقل:

الصورة 22 .21. تهيأة مصدر البيانات.

سنضع ايضاً شرط ان للمستخدم الحق في اظهار البضائع تحت سعر معين فقط، لذا سنضيف متغيراً ونحدد نوعه بأنه Control ونضع اسم مربع النص بالشكل التالي:

الصورة 22 .22. اضافة شروط على البيانات.

فقط يمكنك تجربة ال Query ، اضغط انهاء.

الآن سنقوم بالعودة إلى ال DataGrid، قم باختيار DataSource لها ليكون ال SqlDataSource الذي قمنا بانشاءه منذ قليل بالشكل التالي:

الصورة 22 .23. تعيين مصدر البيانات الخاص بال Data Grid.

وفقط ، قم بتجربة البرنامج ، قم باختيار ارقام مختلفة والضغط

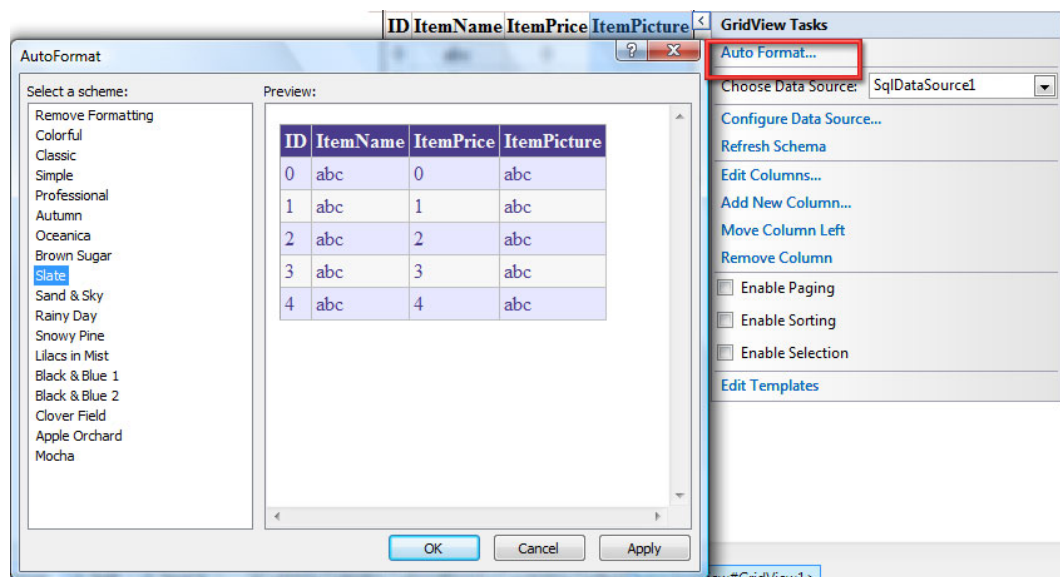
على زر Enter ، الناتج سيكون شيئاً مشابهاً لهذا:

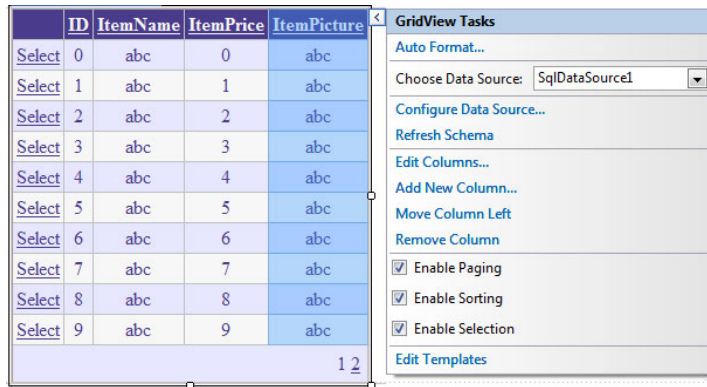
ID	ItemName	ItemPrice	ItemPicture
1	Car Games	50	car.jpg
2	Orange	10	orange.jpg
3	Apple	15	apple.jpg

الصورة 22. 24. عرض البيانات على ال Data Grid.

خصائص اضافية

سنتحدث الآن عن بعض الخصائص الاضافية المرفقة بهذه الأداة ، اولها هي المظهر.

يمكنك تعديل المظهر من خلال CSS خاص، ويمكنك الاختيار بين الموجود بالشكل التالي :

الصورة 22. 25. تهيئة طريقة عرض ال Data Grid باستعمال ال Css.

النقطة الثانية ، هي السماح بالعمليات المختلفة على أداة GridView، يمكنك السماح بوجود الصفحات ، والاختيار والترتيب ايضاً:

الصورة 22. 26. اختيار العناصر في ال **DataGrid** على عكس ال **GridView**.

الناتج سيكون شيئاً بالشكل التالي:

	ID	ItemName	ItemPrice	ItemPicture
Select	2	Orange	10	orange.jpg
Select	3	Apple	15	apple.jpg

الآن سنجرب وضع حدث للاختيار، يمكنك قراءة السجل المحدد باستخدام:

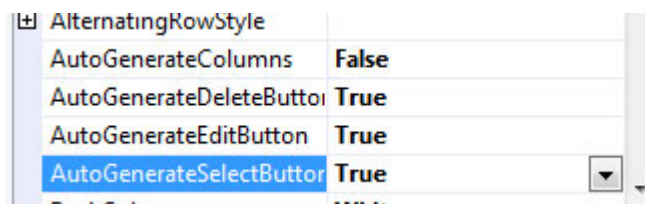
كود C#

```
string something = GridView1.SelectedRow.Cells[0].ToString();
```

كود VB.NET

```
Dim something As String
something=GridView1.SelectedRow.Cells(0).ToString()
```

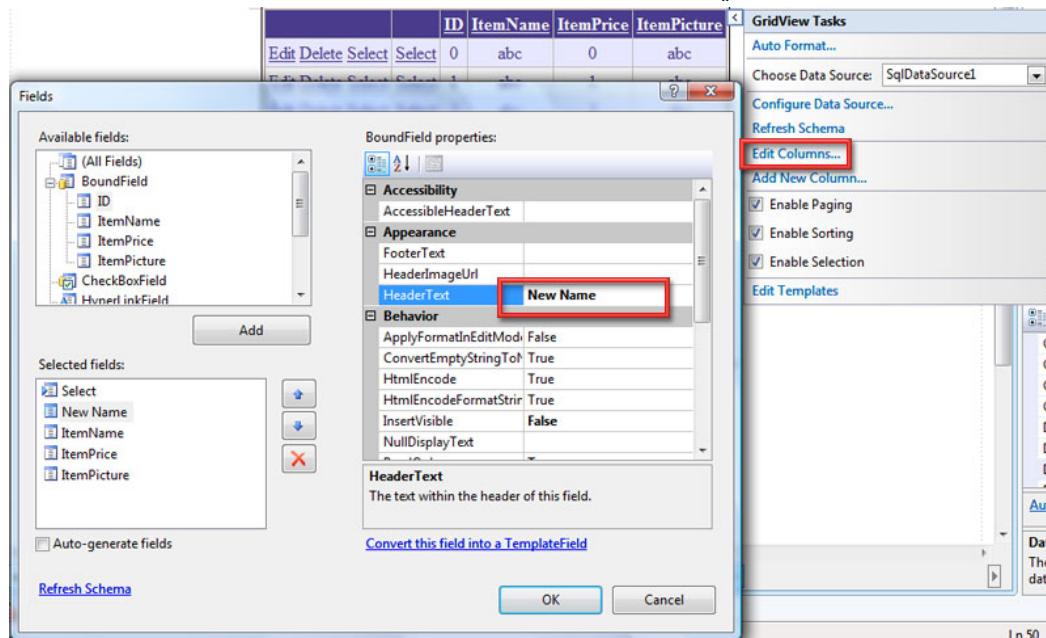
الآن سنجد ضمن الخصائص الخواص التالية:

AlternatingRowStyle	
AutoGenerateColumns	False
AutoGenerateDeleteButton	True
AutoGenerateEditButton	True
AutoGenerateSelectButton	True

تسمح لنا هذه الخصائص بعمليات الحذف والتعديل والاضافة ، بالشكل التالي مثلاً:

	ID	ItemName	ItemPrice	ItemPicture
Edit Delete Select Select	1	Car Games	50	car.jpg
Update Cancel	2	Orange with apple	10	orange.jpg
Edit Delete Select Select	3	Apple	15	apple.jpg
Edit Delete Select Select	4	Fish	85	fish.jpg

الصورة 22. 27. امكانيات العرض الاضافية التي توفرها الخصائص AutoGenerateEditButton ...

هناك ايضاً ادوات اخرى مثل FormView و DetailsView والتي تستخدم ايضاً للعرض وخلافه.

لكل هذه الادوات ، يمكن التعديل في خصائص الأعمدة واسمائها حسب ما تريد بالشكل التالي مثلاً:

الصورة 22. 28. تعديل خصائص الأعمدة.

كانت هذه جولة سريعة في عالم قواعد البيانات مع ASP.net، باقي التفاصيل تجدها في الدروس الاساسية لل ADO.net ...

10. WAP

في هذا الدرس، سوف نتعرف سوية على التقنية المعروفة باسم Wireless Application Protocol والتي تعرف اختصاراً باسم WAP، كما سنتطرق في الجزء الثاني من هذا الدرس إلى كيفية التعامل مع هذه التقنية من خلال net.

10 .1. ماهي WAP ؟

تطبيقات الموبايل Mobile Application

لم يعد الموبايل وسيلة اتصال بسيطة لاجراء المكالمات الهاتفية مثلما كان الهدف منه وقت ظهوره ، إنما تطور الأمر الآن ليصبح الموبايل وسيلة خدمية وترفيهية وتعليمية أيضاً ، ولم يعد الموبايل يستخدم من أجل المكالمات بل أصبح يستخدم في الدخول على الانترنت واستخدام التطبيقات المتقدمة والملتيميديا وغيرها .

ولهذا السبب وغيره ظهرت مصطلحات جديدة لتدل على تطبيقات الموبايل وبرامجه ونظم التشغيل الخاص به، وكان من ضمن هذه المصطلحات مصطلح WAP مرادفاً لمصطلح WEB على الحاسبات الشخصية والذي يرتبط بعالم الإنترنت وخدماته .

ما هو ال WAP ؟

كما اسلفنا سابقاً فكلمة WAP هي اختصار لكلمة Wireless Application Protocol، وكما هو واضح من الاسم فإن هذا يعني ان WAP هي معيار أو Standard عالمي يهدف إلى ربط أجهزة الهواتف النقالة Mobiles بالإرتباط بالإنترنت ، وتم تطويره في النصف الثاني من تسعينات القرن المنصرم .

وقد جاء هذا البروتوكول الموحد ليلم شتات أفكار شركات المحمول الكبرى والتي كانت كل منها تعمل منفردة وفي اتجاه مختلف عن الآخرين .

كيف تعمل WAP ؟

تتميز WAP عن WEB بأنها تعمل في اطار امكانيات الهواتف المحمولة ، وهذا ما يستلزم بالضرورة تصغير حجم البيانات والتعامل مع ضعف السرعة وضعف قدرات المعالجات الخاصة

بالأجهزة المحمولة مقارنة بالأجهزة الشخصية وصغر حجم الشاشة التي يتم عرض البيانات من خلالها أيضاً .

ومن خلال تقنية ال WAP تم لم شمل شتات شركات المحمول الكبرى والتي كانت كل منها تسير فريدة في اتجاه مختلف سعياً لادخال خدمات الانترنت على أجهزتها المحمولة .

وتبدأ قصة ال WAP من خلال طلب العميل ل URL معين لصفحة ما أو لملف وخلافه ، وفي حالة طلب صفحة فإنه يتم ترجمة HTML إلى WML وهي طريقة عرض مناظرة لل HTML يتم استخدامها ضمن بروتوكول ال WAP وسوف نشرحها في الفقرة التالية .

والجدير بالذكر أن بعض الهواتف المحمولة أصبحت تدعم HTML أيضاً ...

10 .2. WML

هي طريقة لوصف البيانات مشابهة جداً لطريقة عمل HTML ولكنها على معايير XML، وتختلف عن HTML في عدد من النقاط منها :

- الوسم الرئيسي هو Tag <wml> بدلاً من <html>

- ينقسم المستند إلى بطاقات Cards لتسهيل العرض حيث تبدأ كل بطاقة بـ <card> وتنتهي أيضاً بـ <card/>.

بداية سنعرض مثالاً يوضح كيفية كتابة WML وذلك بالشكل التالي مثلاً - من ويكيبديا - :

كود	WML

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//PHONE.COM//DTD WML 1.1//EN"
"http://www.phone.com/dtd/wml11.dtd" >
<wml>
   <card id="main" title="First Card">
     <p mode="wrap">This is a sample WML page.</p>
</card>
</wml>
```

قراءة القيم المختلفة وتخزينها وعرضها للمستخدم :

بكل بساطة يمكنك تعريف المتغيرات واسناد قيم لها بالشكل التالي :

كود	WML

```
<setvar name="First_Name" value="Ahmed"/>
<setvar name="Age" value="21"/>
```

ولاحقاً يمكنك عرضها بالشكل التالي مثلاً :

كود	WML

```
<p>First Name: $(First_Name)</p>
```

ويمكن ايضاً قراءة قيمة مربع نص أو Select بنفس الطريقة تماماً حيث يتم وضع اسم الأداة بدلاً من اسم المتغير ...

مثال لقراءة قيم أداة اختيار وعرضها للمستخدم – مثال منقول - :

كود	WML

```
<card id="card1" title="Tutorial">
<do type="accept" label="Answer">
<go href="#card2"/>
</do>
<p>
<select name="name">
<option value="HTML">HTML Tutorial</option>
<option value="XML">XML Tutorial</option>
<option value="WAP">WAP Tutorial</option>
</select>
</p>
</card>
<card id="card2" title="Answer">
<p>
You selected: $(name)
</p>
</card>
</wml>
```

10. 3. WAP+ASP.net

إذا قمنا بتحويل التطبيق الذي عرضناه في أول المقالة - المنقول من ويكيبيديا - والذي كان بالشكل التالي :

كود	WML

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//PHONE.COM//DTD WML 1.1//EN"
"http://www.phone.com/dtd/wml11.dtd" >
<wml>
   <card id="main" title="First Card">
     <p mode="wrap">This is a sample WML page.</p>
</card>
</wml>
```

وقمنا بتحويله إلى net. فسوف يكون بالشكل التالي :

كود	ASP.NET

```
<%@ Page Inherits="System.Mobile.UI.MobilePage" Language="C#" %>
<%@ Register TagPrefix="mobile" Namespace="System.Mobile.UI" %>
<mobile:Form runat="server">
<mobile:Label runat="server">
This is a sample WML page.
</mobile:Label>
</Mobile:Form>
```

في حالة رغبت في التعامل بواسطة VB.net غير السطر الأول فقط ليصبح بالشكل التالي :

كود	ASP.NET

```
<%@ Page Inherits="System.Mobile.UI.MobilePage" Language="C#" %>
```

والآن عند طلب الصفحة من قبل العميل، يتم أولاً تحويل الصفحة إلى WML في حالة كان الطلب من جهاز هاتف محمول ، أو يتم التحويل مباشرة إلى HTML في حالة طلب الموقع من جهاز شخصي أو كومبيوتر كفي .

كان هذا أبسط مثال لتطبيق WAB من خلال NET. ، ولكن ما زال بامكانك عمل الكثير في هذا المجال في موضوع برمجة الموقع عموماً ، وفي التعرف على الأجهزة الزائرة وامكانياتها وغير ذلك من الخيارات المتقدمة خصوصاً .

11. Ajax

في هذا الدرس الأخير حول ASP.net سنستعرض سريعاً كيفية الاستفادة من خدمات تقنية الإنترنت المعروفة باسم AJAX من خلال net.

11. 1. أجاكس Ajax

هي اختصار لكلمة Asynchronous JavaScript and XML ، وفي الواقع فهي ليست لغة برمجة جديدة أو تقنية جديدة قدر ما هي استخدام للموارد الموجودة بطريقة أخرى ، وببساطة ، تعتمد اجاكس على تجزيء الصفحة إلى عدة اقسام تتم معالجة كل قسم على حدة ، وفي هذه الحالة فإنه عند طلب العميل لجزء ما لن يكون مضطراً لانتظار تحديث الصفحة بالكامل.

ولعل أشهر أمثلة استخدام AJAX هو البريد الإلكتروني Gmail اضافة إلى النسخة الجديدة من البريد الإلكتروني لل Yahoo و Hotmail .

لمعرفة المزيد ربما تستطيع زيارة صفحة ويكيبديا عن هذه التقنية:

رابط

http://en.wikipedia.org/wiki/Ajax_%28programming%29

كيف أبدأ من خلال .net ؟

قامت مايكروسوفت باصدار عدة نسخ من الأدوات التي تساعدك وتسهل الحصول على المميزات المتاحة في AJAX ، لكنك على اية حال تستطيع القيام بهذا الأمر في أبسط صوره يدوياً ، أو باستخدام ما يسمى باسم Atlas وهي نسخة من مايكروسوفت خاصة ب AJAX .

وللمزيد من التسهيل قامت مايكروسوفت باصدار Asp.net Ajax ToolKit ، وهي مجموعة من الأدوات المختلفة التي يمكن استعراضها من خلال هذا الرابط

رابط

http://ajax.asp.net/ajaxtoolkit/

تستطيع زيارة الموقع وتحميل هذه الأدوات من خلال الرابط:

رابط

http://ajax.asp.net/

كما يوفر الموقع مكتبة ضخمة من المواد التعليمية لهذه الأدوات ول Asp.net عموماً ، يمكن الوصول إليها عبر هذا الرابط:

رابط

http://www.asp.net/learn/default.aspx?tabid=63

يمكنك البدء من خلال هذا الفيديو تحديداً:

رابط

http://download.microsoft.com/download/0/f/6/0f651a0f-6f2b-4497-b061-e1b2825e22e0/MSAJAX-ToDoList-Video.zip

وسيقوم بالشرح منذ البداية عن كيفية استخدام Atlas.

وهناك دروس أخرى لشرح كيفية البدء باستخدام Ajax ToolKit مباشرة منها هذا الفيديو:

رابط

http://www.asp.net/learn/videos/view.aspx?tabid=63&id=75
http://www.asp.net/learn/videos/view.aspx?tabid=63&id=76

والآن ، سنبدأ في عمل تطبيقنا الأول...

جرب عمل مربع نص لتعرض فيه الوقت الحالي ، ايضاً قم باضافة صورة لتستطيع تمميز تحديث الصفحة ، ومن ثم قم بكتابة الكود التالي في زر الأمر

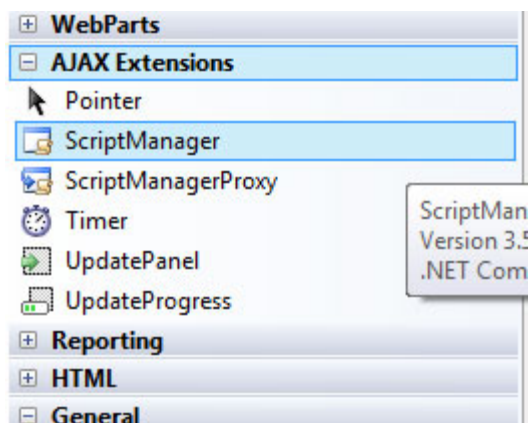
كود C#

```
TextBox1.Text = DateTime.Now.ToString();
```

كود VB.NET

```
TextBox1.Text = DateTime.Now.ToString()
```

هل لاحظت ما يحدث ، بالفعل يتم تحديث الصفحة بالكامل ، الآن سنحاول تطبيق نفس المبدء بحيث لا يتم التأثير سوى على الجزء الذي سيتم التعديل فيه

لذا قم بداية باضافة ScriptManager من ضمن ادوات اجاكس بالشكل التالي:

الصورة 22. 28. تعديل خصائص الأعمدة.

الآن قم بسحب UpdatePanel، وقم بوضع مربع النص وزر الأمر بداخلها مع ترك الصورة فقط في الخارج ، سيكون كود الصفحة بالشكل التالي:

كود	ASP.NET

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default6.aspx.cs"
Inherits="Default6" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>Untitled Page</title>
</head>
<body>
    <form id="form2" runat="server">
    <div>
    <img src="images/food.jpg" />
    <asp:UpdatePanel ID="UpdatePanel1"  runat="server"><ContentTemplate>

        <br />
        <br />
        <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
        <br />
        <asp:Button ID="Button1" runat="server" onclick="Button1_Click"
            Text="ShowTime" />
            </ContentTemplate></asp:UpdatePanel>

    </div>
    </form>
</body>
</html>
```

والآن جرب ، هل لاحظت الفارق فعلاً ؟؟

جميل ، الآن سنحاول تطبيق موضوع آخر ماذا لو افترضنا اننا نريد وضع زر الامر خارج ال update panel بحيث لا يتم تحديثه ، هذا ممكن.

فقط اسحب زر الأمر خارج ال update Panel ، وقم بكتابة الكود التالي داخل ال update panel :

كود	ASP.NET

```
<Triggers>
    <asp:AsyncPostBackTrigger ControlID="Button1" EventName="Click" />
</Triggers>
```

ليصبح الكود كاملاً بالشكل التالي:

كود	ASP.NET

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default6.aspx.cs"
Inherits="Default6" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>Untitled Page</title>
</head>
<body>

    <form id="form2" runat="server">
    <asp:ScriptManager ID="ScriptManager1" runat="server">
    </asp:ScriptManager>
    <div>
    <img src="images/food.jpg" />
    <asp:UpdatePanel ID="UpdatePanel1" runat="server"><ContentTemplate>

        <br />
        <br />
        <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
        <br />

            </ContentTemplate>
            <Triggers>
            <asp:AsyncPostBackTrigger ControlID="Button1" EventName="Click" />
            </Triggers>
            </asp:UpdatePanel>
    <asp:Button ID="Button1" runat="server" onclick="Button1_Click"
            Text="ShowTime" />
    </div>
    </form>
</body>
</html>
```

هل لاحظت الفارق مرة أخرى ؟

عودة لقواعد البيانات

سنعود لتطبيقنا الأخير حول قواعد البيانات ، جرب وضع صورة في الصفحة ، وجرب وضع اي قيمة في حقل السعر الأقصى وقم بالضغط على Enter.

هل لاحظت ما يحدث ، ايضاً يتم تحديث الصفحة بالكامل ، في تطبيقنا التالي سنجعل التحديث يطال ال GridView فقط دون أن تتأثر باقي محتوياته.

الموضوع بسيط جداً كما جربناه ، ضع update panel ، وضع فيها ال GridView ، اضف زر امر ، ومن ثم اضف ال Triggers كما تعودنا في الدرس الماضي ، سيصبح الكود الكامل بالشكل التالي:

ASP.NET كود

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default5.aspx.cs"
Inherits="Default5" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server"><title>Untitled Page</title></head><body>
    <form id="form2" runat="server"><center><div>
        <asp:SqlDataSource ID="SqlDataSource1" runat="server"
 ConnectionString="<%$ ConnectionStrings:ItemsConnectionString %>"
 SelectCommand="SELECT * FROM [Items] WHERE ([ItemPrice] &lt; @ItemPrice)">
            <SelectParameters>
                <asp:ControlParameter ControlID="TextBox1" DefaultValue="0"
Name="ItemPrice" PropertyName="Text" Type="Int64" />
            </SelectParameters>
        </asp:SqlDataSource><br /><br />
        <asp:Button ID="Button1" runat="server" Text="Button" />
 <asp:TextBox ID="TextBox1" runat="server">1000</asp:TextBox>
        <asp:Label ID="Label1" runat="server" Text="البحث عن البضائع تحت
سعر"></asp:Label><br />
        <asp:ScriptManager ID="ScriptManager1" runat="server"
EnablePartialRendering="true">
        </asp:ScriptManager><br />
        <asp:UpdatePanel ID="UpdatePanel1" runat="server" >
            <ContentTemplate>
                <asp:GridView ID="GridView1" runat="server"
AutoGenerateColumns="False" DataKeyNames="ID" DataSourceID="SqlDataSource1"
AllowPaging="True" AllowSorting="True" AutoGenerateDeleteButton="True"
AutoGenerateEditButton="True" AutoGenerateSelectButton="True" BackColor="White"
BorderColor="#E7E7FF" BorderStyle="None" BorderWidth="1px" CellPadding="3"
GridLines="Horizontal" onselectedindexchanged="GridView1_SelectedIndexChanged">
                    <FooterStyle BackColor="#B5C7DE" ForeColor="#4A3C8C" />
                    <RowStyle BackColor="#E7E7FF" ForeColor="#4A3C8C" />
                    <Columns>
           <asp:CommandField ShowSelectButton="True" />
           <asp:BoundField DataField="ID" HeaderText="ID" InsertVisible="False"
                    ReadOnly="True" SortExpression="ID" />
           <asp:BoundField DataField="ItemName" HeaderText="ItemName"
                    SortExpression="ItemName" />
           <asp:BoundField DataField="ItemPrice" HeaderText="ItemPrice"
                    SortExpression="ItemPrice" />
           <asp:BoundField DataField="ItemPicture" HeaderText="ItemPicture"
                    SortExpression="ItemPicture" />
                    </Columns>
 <PagerStyle BackColor="#E7E7FF" ForeColor="#4A3C8C" HorizontalAlign="Right" />
 <SelectedRowStyle BackColor="#738A9C" Font-Bold="True" ForeColor="#F7F7F7" />
 <HeaderStyle BackColor="#4A3C8C" Font-Bold="True" ForeColor="#F7F7F7" />
 <AlternatingRowStyle BackColor="#F7F7F7" />
                </asp:GridView>ContentTemplate>
            <Triggers><asp:AsyncPostBackTrigger EventName="Click"
ControlID="Button1" /> </Triggers>
        </asp:UpdatePanel><br /><br /><br /></div><center></form></body>
</html>
```

الآن فقط قم بتجربة الصفحة، ولاحظ الفارق.

AJAX Toolkit

توفر مايكروسوفت مجموعة جميلة من الأدوات التي تساعدك على تطبيق مبادئ AJAX، يمكنك استعراض الادوات المقدمة هنا:

رابط

http://www.asp.net/ajax/ajaxcontroltoolkit/samples/

يمكنك معرفة كل شيء عن اي واحدة فيهم عن طريق الفيديوهات التعليمية ، تجدها هنا:

رابط

http://www.asp.net/ajax/ajaxcontroltoolkit/

خاتمة الكتاب

إلى هذا الحد نكون قد وصلنا لنهاية جولتنا مع عالم net 2008. والذي يحتوي على النقاط الأساسية في عالم net 2008. ، أرجو من الله أن يكون مفيداً وأن أكون قد ساهمت في زيادة المعرفة ولو بشيء يسير .

يمكنك مواصلة الإطلاع على آخر نسخة الكتاب عن طريق صفحة الكتاب على مدونتي الإلكترونية:

رابط

http://www.AhmedGamal-Technical.blogspot.com

وكما ذكرت في البداية أعود وأكرر في النهاية ، فضلاً لو وجدت أي خطأ في الكود أو في المعلومة أو حتى خطأ إملائي فسأكون في غاية السعادة لو راسلتني به على بريدي الإلكتروني، ايضاً سأكون سعيداً لو كان لديك اقتراحات لتطوير وتحسين النسخ المستقبلية من هذا الكتاب .

أكرر شكري مرة أخرى لكل من ساهم أو ساعد أو دعمني في كتابة هذا الكتاب.

والحمد لله رب العالمين .

وصلى الله على نبينا محمد وعلى آله وصحبه أجمعين .

المــراجـــع

- كتاب Pro C# 2008

Pro C# 2008 and the .NET 3.5 Platform, Exploring the .NET universe using curly brackets.

Aouther: Andrew Troelsen.

Edition: Fourth Edition

- MSDN

MSDN – Microsft Developer Network,

رابط

http://msdn2.microsoft.com/en-us/default.aspx

- مقالات متفرقة من منتدى فيجوال بيسك للعرب .

رابط

http://www.vb4arab.com

- مقالات متفرقة من منتدى الفريق العربي للبرمجة .

http://www.arabteam2000-forum.com/